Multi-step ahead time series prediction

Fouad Bahrpeyma

B.Sc. in Computer Engineering

M.Sc. in Artificial Intelligence

Thesis Submitted for the Award of Doctor of Philosophy (Ph.D.)

Faculty of Engineering and Computing, School of Computing

Dublin City University

Supervisors: Dr. Andrew McCarren and Prof. Mark Roantree

December 2020

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:_____ ID No.: 16213688 Date: December 2020

Acknowledgements

I am deeply grateful to Dr Andrew McCarren and Prof Mark Roantree for there help, guidance, encouragement and support over the last four years.
I would also like to thank the School of Computing, the Kepak Group, the ISIGHT Centre for Data Analytics and Science Foundation of Ireland for providing me with the necessary opportunity and funding to undertake this research. Finally, my great appreciation and love to my parents Mohammad-Reza and Sima for their support and understanding over the years.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Explanation
ACF	Autocorrelation Function
AIC	Akaike information criterion
ANOVA	Analysis Of Variance
AR	Autoregressive
ARCH	Autoregressive Conditional Heteroscedasticity
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	Autoregressive Integrated Moving Average with Explanatory Variable
ARMA	Autoregressive Moving Average
AUC	Area Under the Curve
CH	Conditional Heteroscedasticity
CR	Coverage Rate
DFA	Detrended Fluctuation Analysis
DA	Discriminant Analysis
DIR	Direct strategy
DirMo	Direct Multiple Outputs
DirREC	Direct Recursive
DT	Decision Tree
ED	Early Detection
ES	Exponential Smoothing
fBm	fractional Browning motion
fGn	fractional Gaussian noise

Continued on next page

Continued from previous page

Abbreviation	**Explanation**
FI	Fisher Information
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GAN	Generative Adversarial Net-works
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GoD	Gaussianity of Differences
HMM	Hidden Markov model
HIVE-COTE	Hierarchical Collective of Transform based Ensembles
LRD	Long Range Dependence
LS-SVR	Least Square-Support Vector Regression
LSTM	Long Short Term Memory
MA	Moving Average
mBm	multi-fractional Browning motion
MDN	Mixture Density Network
MIMO	Multiple Inputs Multiple Outputs
ML	Machine Learning
MCCV	Monte Carlo Cross Validation
MRFA	Multi Resolution Forecast Aggregation
MSAP	Multi Step Ahead Prediction
MSE	Mean Squared Error
nMSE	normalized Mean Squared Error
NN	Neural Networks
OSAP	Single Step Ahead Prediction
PACF	Partial Autocorrelation Function
PCA	Principal Component Analysis
PDF	probability distribution function
PH	Prediction Horizon

Continued on next page

Continued from previous page

Abbreviation	**Explanation**
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
REC	Recursive strategy
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic Curve
RoI	Resolution of Impact
RW	Random Walk
SMOTE	Synthetic Minority Oversampling TEchnique
SOM	Self-Organizing Map
SVM	Support Vector Machine
SVR	Support Vector Regression
SW	Sliding Window
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TU	Time Unit

Abstract

Fouad Bahrpeyma

Multi-step ahead time series prediction

Time series analysis has been the subject of extensive interest in many fields of study ranging from weather forecasting to economic predictions, over the past two centuries. It has been fundamental to our understanding of previous patterns within data and has also been used to make predictions in both the short and long term horizons. When approaching such problems researchers would typically analyze the given series for a number of distinct characteristics and select the most appropriate technique. However, the complexity of aligning a set of characteristics with a method has increased in complexity with the advent of Machine Learning and the introduction of Multi-Step Ahead Prediction (MSAP). We examine the model/strategy approaches which are currently applied to conduct multi-step ahead prediction in time series data and propose an alternative MSAP strategy known as Multi-Resolution Forecast Aggregation.

Typically, when researchers propose an alternative strategy or method, they demonstrate it on a relatively small set of time series, thus the general breath of use is unknown. We propose a process that generates a diverse set of synthetic time series, that will enable a robust examination of MRFA and other methods/strategies. This dataset in conjunction with a range of popular prediction methods and MSAP strategies is then used to develop a meta learner that estimates the normalized mean square error of the prediction approach for the given time series.

Chapter 1

Introduction

This dissertation is about time series predictions, the difficulty in selecting appropriate models to address particular problems and time series data, and being able to ensure robust evaluations for new predictive models. In this Chapter, we present the background and motivation for the research proposed in this thesis. In particular, we will outline the main research questions and highlight the main contributions that we believe will emerge form our work. In §1.1, we present an introduction to Time Series predictive algorithms and highlight some use case applications. In §1.2, we motivate the issues with time series predictions and present our problem statement. In §1.3, we present the research questions that we address in this work and articulate our contributions. Finally, in §1.4, we summarize the Chapter and outline the structure of this dissertation.

1.1 Time Series and their Prediction Strategies

A time series can be considered to be a sequenced series of data points that correspond to measurements of an object, phenomena or signal that are taken at equidistant time points [47]. In this dissertation we focus on equidistant time series which is consistent with the majority of the literature in the time series domain. Typically, they will correspond to a single series or signal and can be defined as a vector

$x(t); t = 0, 1, 2, ...$, where t is the time lapse and x is a random variable. The samples taken from the time series process are arranged in the original chronological order of occurrence. Researchers have used exogenous variables to predict future values of the series with varying levels of success. However, there is a considerable body of work which advocates the use of the *series itself* to predict future values. The advantage of such an approach is that identifying the relevant input/exogenous variables is non-trivial and the use of the signal itself is based on the principle that there is a considerable volume of information in past values of the series [66].

Times series methods have been applied and studied in many disparate disciplines, which include econometric models in financial decision making [251], signal processing applications in physics [57], and to anticipate weather and climate changes in climatology [82]. In addition to the subject type, time series methods have also been applied to differing data outcome types. For example, time series **classification** was used to analyse EEG signals in [234] and activity recognition in [125]. While time series **clustering** was implemented on active community detection in mobile networks in [141] and trajectory clustering to improve query evaluation performance in [141]. The extensive use of time series analysis in such a wide spectrum of applications has led to the development of a wide range of methods and prediction strategies.

Traditionally, in time series prediction studies, the general approach has been to use historical data to predict a single time point in the future, known as a One Step Ahead Prediction (OSAP) strategy [277]. Most time series prediction approaches require stationarity as a pre-requisite; a stationary process has a constant mean and a constant variance. In 1905, Pearson identified a class of prediction models known as Random Walks (RW), that are based on the assumption that at each time point the time series takes an identically independently distributed (iid) random movement from the previous value [193]. RW is the simplest form of non-stationary time series model and for the time series x_t, is described in Eq. 1.1, where w_t is an *iid* normal variable i.e., a zero-mean process with a constant variance σ^2.

$$x_t = x_{t-1} + w_t \tag{1.1}$$

In practice, w_t is usually assumed to be a Gaussian white noise or $w_t \sim N(0, \sigma^2)$ (σ is the variance), which is a special case of the Autoregressive (AR) processes [249]. Eq. 1.1 emphasizes that an RW assumes that all information regarding the *future* of the signal exists in the available data. Almost two decades ago, RWs were the predominant linear models used for time series data analysis, and especially the case in financial applications [3]. Various adjustments and alterations have also been made to RW in the literature such as RW with drift and error correction terms [3]. However, RW has been shown to be an inappropriate choice for capturing non-linear patterns [3, 136].

RW is similar to another type of stochastic random processes known as the *Markov Chain*. Markov Chain is a stochastic process where the behavior evolves according to an index $t \in T$, in a random manner, and is characterized by the Markov property. A process has the Markov property if its n^{th} state only depends on its $(n-1)^{th}$ state. Markov models have also been used for time series prediction, such as in [105] where the use of Markov prediction models in stock models has been studied.

Another class of predictive analyses was initiated in 1944 by Brown known as *Exponential Smoothing* (ES) models [92]. The ES model has been widely accepted in the time series community because of its ease of use. Winter [270] then presented a similar method known as the *Holt-Winters* model, which comprised additional steps for handling the concepts of *additive trends* and *seasonality*. Trend carries the information associated with long term or low frequency behavior of the series. Many time series exhibit a regularly repeating pattern known as *seasonality*, often under the influence of external periodic drivers such as seasons, weather or holidays.

Exponential smoothing (ES) is based on the belief that *recent* observations provide more information than *older* observations. ES assigns exponentially decreasing weights to the lags as the lag gets older and then incorporates a weighted averaging to obtain the forecast [142]. ES performs well when variables change over time

slowly and can be described by Eq. 1.2, where α is the weight and $0 \leq \alpha \leq 1$.

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + ... \tag{1.2}$$

With $\hat{y}_{t+1} = l_t$, Eq. 1.2 is an expansion of a simple smoothing approach, shown in Eq. 1.3:

$$l_t = \alpha y_t + \alpha(1-\alpha)l_{n-1} \tag{1.3}$$

where l_0 is calculated by Eq.

$$l_0 = \bar{y} = \frac{\sum_{t=1}^{n} y_t}{n} \tag{1.4}$$

A larger α value gives a higher weight to the original values and thus, a more fluctuated curve is obtained; while a smaller α results in a smoother signal. An advantage of ES is that it devotes a greater significance to recent observations and thus, with the smoothing technique, random fluctuations will have less impact on the accuracy of the prediction results. The main shortcoming with ES approaches derives from their initial assumption about the model that the fluctuations in the given time series should lay around a fixed level or change slowly. In the presence of a significant trend or seasonality, even an adaptive ES model fails to obtain accurate forecasts [128]. Autoregressive (AR) models were introduced as one of the first types of time series stochastic modeling methods [285], and are built by modelling the current time points on lagged versions of themselves. The AR model with order p is defined in Eq. 1.5.

$$\hat{y}_{t+1} = c + \sum_{t+1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{1.5}$$

where $\phi_1, ..., \phi_p$ are the model parameters, c is a constant and ϵ_t is white noise.

In 1937, Slutzky presented the concept of a Moving Average (MA) model [235], which made a significant contribution to the statistical time series analysis. The MA model with order q is defined in Eq. 1.6.

$$\hat{y}_{t+1} = \mu + \sum_{i+1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \tag{1.6}$$

where $\theta_1, ..., \theta_q$ are the model parameters, μ is the mean of the series and $\epsilon_1, ..., \epsilon_q$ are the white noise error terms.

A year later, Wold [271] put the AR and the MA models together and introduced the ARMA model which could describe a large class of stationary time series processes. Wold's method was not properly implemented until technology provided researchers with the computing power to optimize parameters for the AR (Eq. 1.5) and the MA (Eq. 1.6) components of the ARMA model in the 1960s, shown in eq. 1.7.

$$(1 - \sum_{i=1}^{p'} \varphi_i L^i) y_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \varepsilon_t \tag{1.7}$$

where L^i is the shift operator.

Some studies have used decomposition techniques such as Wavelet transforms to further improve prediction performance. For example, ARIMA has been used with wavelet transform in [144] to forecast metal prices. Also, ES has been combined with the wavelet transform to predict sediment load in [232].

Multi-Step Ahead Prediction. More recently, approaches have been proposed which predict multiple future time points and these are known as *Multi-Step Ahead Prediction* methods (MSAP) [28]. MSAP approaches have had a variety of applications, in areas such as wind speed prediction [261], Hydrological time series prediction [138] and crude oil prices prediction [61]. In MSAP studies, several strategies such as the Direct and Recursive approaches have been proposed. The existing MSAP strategies including the Direct and the Recursive strategies are explained

in details in Chapter 2, section 2.2. Both can be implemented using different machine learning techniques, such as Neural Networks (NNs), Recurrent Neural Networks (RNNs) and Support Vector Regression (SVR). Throughout the literature, researchers have implemented a wide range of both prediction methods and strategies. The success of each method and prediction strategy has been shown to be subject to the suitability of each approach to accommodate characteristics or features within the time series. A good example of this is the ARIMA model which was developed in the 1970s by Box and Jenkins [49]. This method relies on the assumption that there should be a constant variance in the series under examination and if this property is not present, can undermine future predicted outcomes. Even modern machine learning approaches can have their predictive power reduced by the characteristics of a particular dataset. Neural Networks are an example where long term memory in the data weakens the algorithm's predictive powers [163].

Researchers incorporate an approach known as the Sliding Window (SW) to convert a time series prediction problem to a classical supervised learning problem so that machine learning techniques can be used for implementing time series prediction. SW is a fixed length frame sliding over a time series, each time recording the covered values as a new sample required for training machine learning models. In recent years LSTM [97], RNN [178] and other machine learning approaches have received a lot of attention in time series studies [203]. However, little attention has been given to the appropriateness of these methods in improving MSAP performance and the time series characteristics where they work best. Typically, in OSAP one finds that certain approaches are appropriate when the data source has certain characteristics or features. In order to examine the range of appropriate characteristics for a proposed method researchers have implemented their approaches on either well-established data repositories such as Kaggle [108] or have attempted to generate synthetic data. Regardless of the strengths and weaknesses of each strategy or model, choosing the appropriate approach can be challenging. The literature states that there is no universal approach/strategy that will outperform others when making predictions, and this concept is known as the 'No free lunch" theorem [272]. There have been at-

tempts in OSAP, such as [262] and [156], to build recommender systems that relate time series features/characteristics to method performance. This process is known as prediction method selection or more recently, *meta-learning*, where a machine learning technique is trained to recommend the appropriate method based on a set of features [155]. Meta-learning for OSAP methods is gaining interest and has been applied in a number of studies such as [262] and [29]. However, Meta-Learning in MSAP has not received the same attention in the literature. This presents an interesting opportunity for MSAP researchers.

1.2 Problem Statement

Times series analysis has a long history in Mathematics, Statistics and Econometrics. Typically, the historical applications have been implemented with OSAP strategies and the introduction of MSAP has shown some promise. One of the major challenges that currently exists in time series prediction methods can be attributed to the error accumulation that occurs when predicting forward. Typically, when using a OSAP strategy, predicted values incorporate components of previous values such as in the ARIMA model in Eq. 1.8; where L is the lag operator, d is the order of differencingφ_i are the autoregressive parameters, θ_i are the moving average parameters, and ε_t are the error terms. Eq. 1.8 is an extension of Eq. 1.7 with an additional differencing parameter φ_i.

$$(1 - \sum_{i=1}^{p} \varphi_i L^i)(1 - L)^d X_t = \delta + (1 + \sum_{i=1}^{q} \theta_i L^i)\varepsilon_t \tag{1.8}$$

As one moves forward in the prediction horizon the prediction errors will steadily increase using a model such as that described in eq. 1.8. This is primarily due to the use of predicted values as lagged inputs in the model. Reducing the reliance on the predicted lagged components should in theory reduce the predictive error of future models; Some MSAP approaches such as the direct strategy (defined in Chapter 2, section 2.2) were presented to reduce this reliance.

This strategy is known as the recursive strategy, and its main purpose is to retain the autocorrelation structure (or sequential correlations) of the time series. An alternative approach is to simply build a separate model for each of the future steps using the actual lags for time points going back further in time, and is known as the direct strategy. This will eliminate the need for the consecutive repeats of OSAP over previously predicted values, but there will be a loss of information due to the increased time gap between the predicted and the predictor time points. Several new approaches have emerged each presenting a new theoretical basis to mitigate the error accumulation or the intermediate information loss problem [244]. Researchers incorporate an approach known as the Sliding Window (SW) to convert a time series prediction problem to a classical supervised learning problem so that machine learning techniques can be used for implementing time series prediction. SW is basically a fixed length window sliding over a time series, each time recording the covered values as a new sample required for training machine learning models. When attempting time series prediction, the choice of approach can often depend on the characteristics within the data. The direct strategy, explained in section 2.2, for example will be more relevant when there is long term memory in the data. The results from the proposed MSAP stress test could then be compared to those generated from other methods that are used regularly in the literature.

1.3 Research Questions and Contribution

We now proceed to articulating the research questions addressed in this research which are across 3 broad topics: improvement of existing MSAP methods; a robust validation framework for MSAP research; and the development of a pre-processing step in the application of MSAP models, to assist the researcher in pre-selecting good candidate models for time series prediction.

Research Question 1: Improving MSAP Models. Direct and recursive prediction strategies, which are explained in detail in section 2.2, are well established

techniques when attempting multi-step ahead predictions. Researchers in recent studies have predominantly focused on combined approaches to avoid the exclusion of *sequential correlation* in the direct strategy, or the *accumulation of error* in the recursive strategy. This research question examines how the integration of metrics derived from multi resolutional sliding windows together with a recursive MSAP strategy can be used to reduce error accumulation. Therefore, this research question can be stated as follows:

- How can the integration of metrics derived from multi-resolutional sliding windows together with a recursive MSAP strategy can be used to reduce error accumulation?

Research Question 2: More Robust Validation. A more robust evaluation for new time series methods requires high volumes of data to ensure a high level of rigour in testing. However, for many researchers, the availability of appropriate time series repositories for the given task presents a barrier to this type of robust evaluation. The term *Diversity* refers to the absence of focus on specific domain-specific features in the universe of discourse.

A feature is any single value obtained from applying a test to the time series. Also, the feature space is a multi-dimensional space in which each dimension corresponds to a separate features. This research question can be stated as follows: With the goal of constructing significant numbers of time series, how can a dataset generation strategy be developed to provide the following outputs?

- A dataset that can cover the potential feature space required to robustly test time series algorithms
- Metrics that capture the diversity of the complete dataset generated.

Research Question 3: Meta-Learning. When attempting to model a time series

dataset, researchers are often unable to identify the approach (model or algorithm) that is most suitable for the proposed dataset. In this research question, we attempt to connect the most appropriate technique from a list of common time series MSAP approaches with the synthetic datasets generated as part of research question 2, using features extracted from the generated time series. From this dataset, we will build a machine learning model that will allow researchers to identify potential candidate algorithms. Because we use time series features, we can use our analysis to draw conclusion about real time series. Therefore, the research question can be stated as follows:

- How can we build a machine learning model, using the time series generated in research question 2, that will allow researchers to identify potential candidate algorithms?

1.3.1 Contribution

The primary contribution from this thesis are associated with the research questions outlined above and they are as follows:

- The initial challenge was to develop a MSAP strategy that would potentially reduce the cumulative error that occurs in many competitor strategies. This work was presented in [21] and demonstrated a positive outcome. We also determined that there is a significant improvement when the proposed strategy is integrated with a Recurrent Neural Network.
- In order to test the strategy outlined above, we developed a synthetic data generation approach that creates a uniquely *diverse* test dataset. When applied to a proposed algorithm or strategy, researchers can use the generated datasets to robustly test new time series models.
- We show how meta features generated from the characteristics of time series can be used with a Random Forest regression model to estimate the normalized mean square error for a particular set of algorithms. This approach allows

researchers to narrow down the field of potential candidate algorithms for the proposed dataset, with a significant benefit of reducing the volume of research and experimentation that is generally required during a time series analysis.

1.4 Summary

In this Chapter, we presented a brief overview of the research area and the challenges faced in Multi-Step Ahead Prediction. We have also outlined the research questions and contribution for this thesis. We now provide an outline structure for this dissertation and describe the main goal of each Chapter.

In Chapter 2, we present our literature review for this research. In particular we focus on the application of techniques that are currently being applied in MSAP, where they are used and the weakness that have been encountered with them. In Chapter 3, we give a high level description of our proposed methodologies and explain how the different steps combine to deliver our overall solution. The 3 main contributions in this research are covered in the next 3 Chapters. Chapter 4 is used to describe our proposed MSAP strategy. In Chapter 5, we present the synthetic data generation strategy and show how our evaluation ensures that the overall dataset is diverse and so does not favour any single time series model. In Chapter 6, we introduce our meta learning approach that uses the characteristics of the generated time series data as inputs to a regression model that predicts the normalised mean squared error of a proposed technique. In Chapter 7, we present the evaluation, findings and insights of our research. Finally, in Chapter 8, we present a summary and discussion on future research in the area.

Chapter 2

Related Research

In this Chapter, we present our review of the literature which cuts across a number of topics and important issues in time series prediction research. In §2.1, we present a generalised discussion on time series methods in order to introduce some of the more general open problems. We then examine multi-step ahead prediction strategies in §2.2 as these form a core part of our research. The lack of time series data to robustly evaluate prediction algorithms led a number of researchers to develop synthetic time series. This step, also adopted in our research, has a number of significant issues discussed in §2.3. As part of our study, we discovered how adopting a meta-learning step as part of an overall strategy could yield significant benefits and in §2.4, we present a critique of approaches that adopt a similar strategy. Finally, we present an overall summary of our state of the art discussion in §2.5.

2.1 Time Series Modelling

The choice of time series method with either One Step Ahead Prediction (OSAP) or Multi-Step Ahead Prediction (MSAP) strategies has been shown to have a significant impact on the predictive power of any analysis [205, 244]. There are many methods available to researchers, and each has its own advantages and disadvantages. The choice of method has traditionally focused on the type of data that researchers are

working with or the field of study, and have been classified into two broad groups: parametric and non-parametric. In this section, we will provide a review of research in both groups with the goal of highlighting existing weaknesses and open research problems.

2.1.1 Parametric Time Series Prediction

Time series prediction literature has been influenced by a number of parametric approaches and used to forecast future values for a variety of datasets [176]. The term *parametric* implies that the model follows certain distributional assumptions which when met, can give favourable results [96]. Parametric models have played an important role in the advancement of predictive analyses, and led to the development of some powerful methods such as ARIMA models, and their extensions, [48]. The general properties of parametric models [295] are: stationarity, continuous sample-path, normality, a finite number of parameters and a rigid linear structure. Some methods, such as ARIMA can best be applied to time series data where the time series variance remains constant [287], while others can handle non-constant variance [12]. As far back as 1926, AutoRegressive (AR) models were introduced as the first type of time series stochastic modeling methods [285], and are built by modelling the current time points on lagged versions of themselves. The AR model has been used for load forecasting [162] and river flow forecasting [195]. Later in 1937, the Moving Average (MA) model [235] was presented. The AR and the MA models were put together in [271] and the ARMA model was introduced. In 1970, Box and Jenkins proposed integrating a differencing component with the ARMA model to bring stationarity to some non-stationary time series [49]. This new model was called ARIMA and has been applied extensively in the time series community. In [263], ARIMA was used to predict household food retail prices. This work categorizes 41 different types of food into five groups based on the price movement trends: smoothly rising, rising with fluctuations, stable, horizontal fluctuating and concave. Then, for each category a different ARIMA model was employed. ARIMA was studied in [196] to forecast prices of three different types of palm oil. The results were compared with

those of AR, MA and ARMA models. In [104], ARIMA was conducted to forecast drought using the VTCI index time series, in [297] to forecast food grain prices, and in [242] to forecast Sugarcane yields. An ARIMA model was used to model the test-day milk yields of dairy ewes in [167], and monthly reservoir inflow was forecasted using ARIMA and ARMA in [256]. Forecasting pre-monsoon rainfalls using ARIMA was studied in [191], where the results report that the change in the trend in rainfall which caused non-stationarity in the associated time series was captured properly. ARIMA was also practiced in [278] for forecasting agricultural commodity prices.

Based on the AR and MA models, ARMA can be written as Eq. 2.1:

$$(1 - \sum_{i=1}^{p'} \varphi_i L^i) X_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \varepsilon_t \tag{2.1}$$

With $p' = p - d$ in Eq. 2.1, the $ARIMA(p, d, q)$ model can be generalized to Eq. 2.2:

$$(1 - \sum_{i=1}^{p} \varphi_i L^i)(1 - L)^d X_t = \delta + (1 + \sum_{i=1}^{q} \theta_i L^i) \varepsilon_t \tag{2.2}$$

where L is the lag operator, d is the order of differentiation, φ_i are the autoregressive parameters, θ_i are the moving average parameters, and ε_t are the error terms.

ARIMA models can also be used to model a range of seasonal time series [51]. The Seasonal ARIMA (SARIMA) model is identified by $ARIMA(p, d, q) \times (P, D, Q)_m$, where (p, d, q) are the non-seasonal ARIMA parameters, m is the number of observations per season, and (P, D, Q) are conceptually similar to the non-seasonal components but with the back shift of the seasonal period. SARIMA was used in [148] for traffic flow forecasting, in [24] to predict international tourism demand and in [211] to predict the number of malaria incidents.

The Autoregressive Integrated Moving Average with Explanatory Variable (ARIMAX), is a version of ARIMA that allows the combination of linear regression and ARIMA in order to be able to accommodate exogenous variables in the modeling process [152]. In [35], an ARIMA model was used to predict groundwater levels and then exogenous-variables were added (to build the ARIMAX model) to model the

groundwater levels in response to estimates of land surface recharge.

Limitations with ARIMA approaches. The popularity of the ARIMA family has roots in its flexibility in describing a small class of time series data with simplicity as well as the associated Box-Jenkins methodology for an optimal modeling process [47]. However, a significant drawback with ARIMA models is the assumption of a rigid structure in the underlying time series and it can only model linear relationships between the time series lags which is often unsuited to numerous real-world problems [83]. A variety of non-linear stochastic models have been suggested to tackle this shortcoming [183,287,288]. In comparison with new time series modeling techniques, ARIMA is more *interpretable* but can be less *accurate* for certain data types [280].

2.1.2 Non-Parametric Approaches

Non-parametric models, make no presumptions about the presence of known structures such as the continuity of the sample sequence, stationarity and normality. Thus, they have a greater applicability to a broader range of datasets [72]. The advent of Machine Learning (ML) provided the research community with a new class of non-parametric time series models [96]. ML techniques such as Neural Networks, Recurrent Neural Networks (RNN) or Long Short Term Memory (LSTM) offer a non-parametric approach to model non-linear systems without the need for prior knowledge about the mathematical structure of the system. ML techniques have shown a remarkable capacity to uncover inherent non-linear relationships between time series lags and thus, eliminating the need for the manual specification of the model structure. An important disadvantage with non-parametric models and specially machine learning models is the lack of quantified uncertainty at their outputs. This issue has been an interesting topic of research in the past studies. For example, in [122, 135, 137] studies were undertaken on the estimation of the prediction intervals for Neural Network models. However, no standard method has yet been presented for quantifying forecasts uncertainty in machine learning models and thus,

quantifying forecasts uncertainty is out of the scope of this study. The time series community, adopted ML techniques by converting the sequential supervised learning problem of time series prediction to the classical supervised learning problem that suits ML techniques [78]. However, challenges and issues with both parametric and non-parametric have been reported in the relevant literature suggesting that more considerations need to be taken when choosing an approach [55].

- **Neural Networks**

Neural Networks (NN) are a ML method inspired from the information handling process of the brain's nervous system [73]. A NN is a layered architecture of computational units referred to as neurons. Neurons in the NN are organized in consecutive layers where each neuron is connected to all neurons placed in the previous and the next layer. A NN has an input layer, an output layer and at least one hidden layer. The data records are presented to the NN in the format of input-output pairs, and the network is trained via a gradient-descent based algorithm aiming at simulating the output with respect to the associated inputs [75].

NNs are highly popular among the researchers of various fields because they are theoretically able to model complex nonlinear functions with acceptable approximations. Also, NNs are a type of data-driven approach which can learn the structure of the system solely by observing historical data and do not need to have prior knowledge regarding the design of the system [289]. However, NNs (and other NN based methods) contain complex mappings between inputs and outputs, a feature that is difficult to analyse and understand [227]. Furthermore, NNs use backpropagation, which because of being a gradient descent type of training algorithm, cannot guarantee reaching the global minimum error [202]. Another important drawback with the NN family of methods is the lack of theoretical solutions to specify the optimal number of neurons in the hidden layers [174].

- **Recurrent Neural Networks**

A Recurrent Neural Network (RNN) is a class of NNs that uses recurrent connections to provide the network with memory from a temporal sequence [75]. This memory enhances the flexibility of the network by processing sequential data as the output of the neuron from the previous step y_{t-1}^{Neuron}. This is then used to produce the output for the current step $output_t$. In the training process a back propagation algorithm through time (BPTT) is used to adjust the weights which incorporate the recurrent structure. The architecture of an RNN is shown in Fig. 2.1.

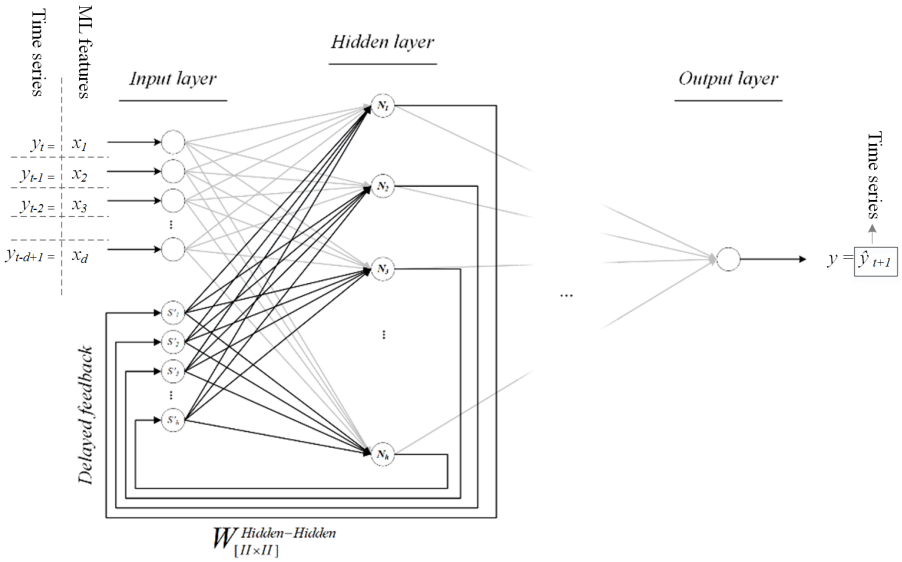


Figure 2.1: Neural architecture of the RNN

In Fig. 2.1, S' are the previous states of the neurons and N_i represents the state of the i^{th} neuron in the hiddden layer. Due to the recurrent feedback, a delay function/operation is introduced to retain the activations until they are processed at the next time step. The behavior of RNNs can be explained as a dynamical system using the formulae in Eq. 2.3, where $X(t)$ and $Y(t)$ are the RNN's input and output vectors, W_{IH}, W_{HH} and W_{HO} are the three connection weight matrices, f_H and f_O are the hidden and output unit activation functions. Note that in Fig. 2.1, if the inputs $x_1, x_2, .., x_d$ include variables other than the time series lags $y_t, y_{t-1}, y_{t-2}, ...$, the non-lag variables are the same as the exogenous variables in an ARIMAX model [152].

$$h(t) = f_H(W_{IH}X(t) + W_{HH}h(t-1))$$
$$Y(t) = f_0(W_{HO}h(t)), \tag{2.3}$$

where $h(t-1)$ is the state matrix of the neurons in the hidden layer in the previous step.

The architecture of a recurrent neuron is shown in Fig. 2.2.

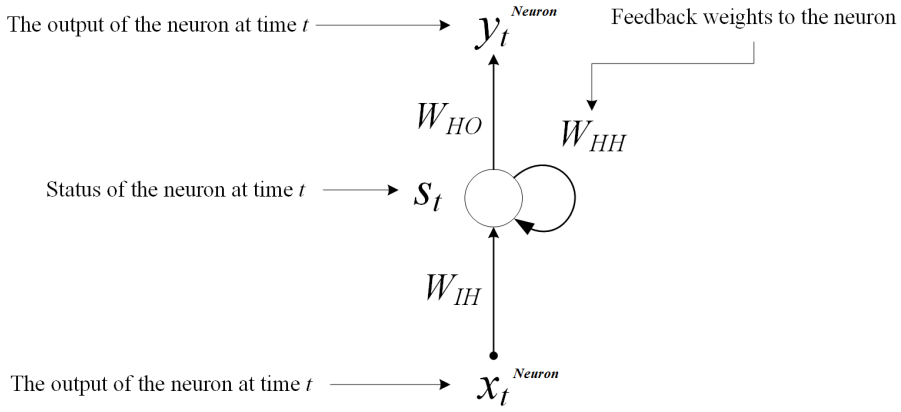


Figure 2.2: A simple neuron with one recurrent feedback

In the information processing mechanism of a recurrent neuron, the temporal information which is referred to as *state* about the subsequent events is processed by a non-linear activation function f_H and the result is fed back to the neuron. In RNNs (unlike NNs), the output of a recurrent neuron depends also on its previous state s_{t-1}, since the recurrent weight W_{HH} informs the neuron about its previous state s_{t-1}. As shown in Fig. 2.2, the recurrent neuron has a cycle (the recurrent feedback) and the neuron's output dynamically changes between time steps. Due to this cycle, the behavior of the neuron is difficult to interpret and the network cannot be trained via traditional backpropagation algorithm [206]. However, the recurrent structure of the recurrent neuron in Fig. 2.2 can be unfolded into a graph which has no cycles and thus the output of the recurrent neuron does not change between the unfolded time steps, which enables the network to be trained using the

backpropagation algorithm. The unfolding operation converts the architecture of the recurrent neuron (shown in Fig. 2.2) to a feedforward architecture, where the degree of recurrence determines the number of hidden layers [173], shown in Fig. 2.3.

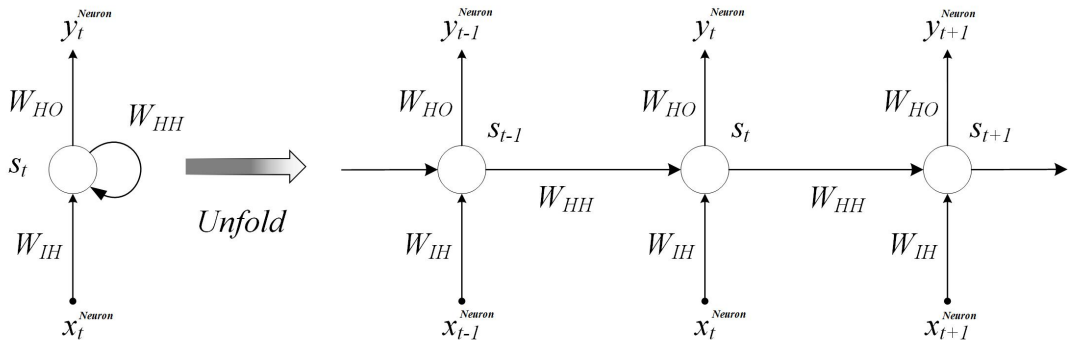


Figure 2.3: Unfolding

As shown in Fig. 2.3, unfolding converts the RNN's architecture into a feedforward structure which has no cycles. Eq. 2.4 describes the variables shown in Fig. 2.3.

$$\begin{aligned} s_t &= f_H(w_{IH}x_t + w_{HH}s_{t-1}) \\ y_t^{Neuron} &= f_O(w_{HO}s_t), \end{aligned} \tag{2.4}$$

where s_t is the state of the recurrent neuron at time t, x_t^{Neuron} is the input to the recurrent neuron at time t, y_t^{Neuron} is the output of the recurrent neuron at time t, f_H is the activation function of the hidden layer, and f_O is the activation function of the output layer. Also, w_{IH}, w_{HH} and w_{HO} are the input weight, the feedback weight and the output weight, respectively.

In theory, a RNN has a memory of unlimited length. However, in practice, the number of recurrences is limited to a few steps [75]. Elman NN (a simple RNN with one hidden layer) was conducted to forecast the COMEX copper spot price in [150], crude oil prices [181], stock returns [219] and wind speed [30].

- **Support Vector Regression**

Support Vector Regression (SVR) is an extension of the Support Vector Machine (SVM) classification methods [189], and both are based on statistical learning theory [258]. SVR follows the same principles as SVM, with a few modifications that enables it to solve regression problems. The basic idea behind SVR is to use a tolerance that maximizes the margin between hyperplanes given in Eq. 2.5.

$$\begin{aligned} y_i &= w.\varphi(x_i) + b + \epsilon \\ y_i &= w.\varphi(x_i) + b - \epsilon \end{aligned} \tag{2.5}$$

These hyperplanes are shown in Fig. 2.4.

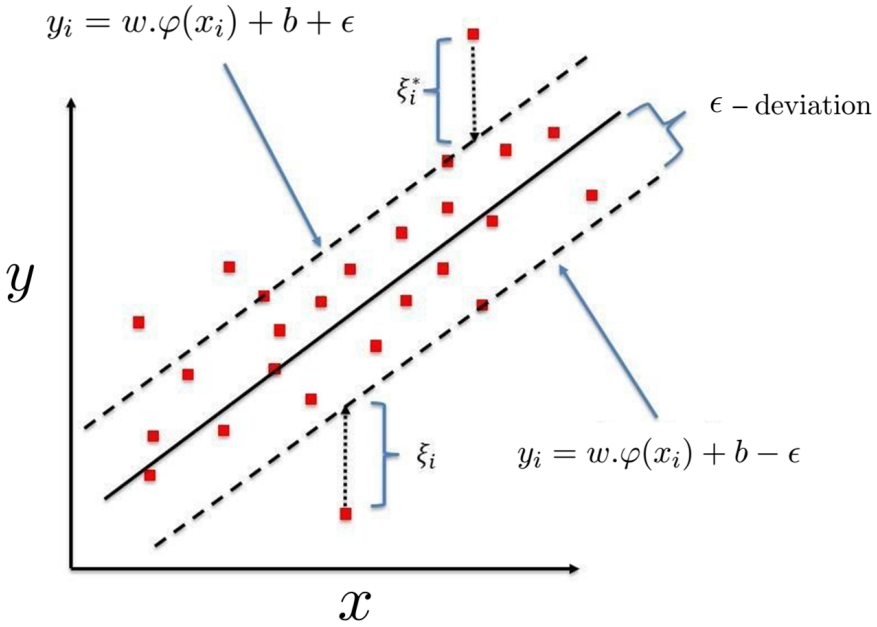


Figure 2.4: SVR hyperplanes

For a time series prediction process, consider $f(x)$ as the function that predicts the value $\hat{x}$ for a prediction horizon Δ_t, based on d past observations (d is the number of time series lags) as shown in Eq. 2.6.

$$\hat{x}(t + \Delta_t) = f(x_{t-d+1}, x_{t-d+1}, ..., x_t) \tag{2.6}$$

Eq. 2.7 emulates the prediction function of Eq. 2.6 as a SVR process:

$$f(x) = (w.\varphi(x)) + b. \tag{2.7}$$

where $\varphi(x)$ is the kernel function and chosen depending on the problem. The radial Basis Function (RBF) and the Polynomial function are the most popular non-linear kernel functions. Note that the number of lags d is another hyper-parameter of an SVR model and its optimal value is obtained through an input selection procedure. By adding a slack variable, the objective of the SVR's learning process is to minimize the cost function given in Eq. 2.8.

$$Min\frac{1}{2}\left\|w\right\|^2 + C\sum_{i=1}^{n}|\xi_i|, \tag{2.8}$$

with the following constraint:

$$|y_i - w_i x_i| \leq |\xi_i|, \tag{2.9}$$

In Eq. 2.7, if the data is linear then φ is a linear function. Otherwise, $\varphi(.)$ is required to project the data into a higher dimension space which is appropriate for a linear regression model [189]. The goal of the process is to find an optimal set of weights w and the threshold b. A linear data is a data that is generated by a linear system and a system is linear when its output Y^{system} is created by a linear function of its inputs X^{system} [273], as shown in Eq. 2.10.

$$Y^{system} = W.X^{system} + B \tag{2.10}$$

where, W is a fixed vector and B is the intercept.

SVR has the advantage that the computational complexities of the SVR process do not depend on the dimensionality of the problem space [16]. However, an important disadvantage of SVR is that increasing the number of training samples causes the training time to grow exponentially [274]. It is hard to determine the trade off

between over-learning and under-learning for the SVR model, since the performance of an SVR model highly depends on the choice of the kernel function ψ and the fine tuning of the hyper-parameter, which is a difficult task [25].

- **Long Short-Term Memory**

In theory, the recurrent structure of RNNs permits the network to incorporate long term memory in a a temporal sequence. However, when using gradient based training methods and the backpropagation algorithm, RNNs usually fail in practice as they suffer from both vanishing and exploding gradients. The Long Short Term Memory network (LSTM) is a variant of RNNs which is capable of handing long term dependencies and thus, enables the network to handle vanishing/exploding gradients, [118]. The prototype Long Short-Term Memory (LSTM) has an input gate, a forget gate, a memory cell, and an output gate. The forward pass in the training process of an LSTM with one forget gate incorporated the functions presented in Eq. 2.11.

$$\begin{aligned}
\text{Forget gate activation} \quad & f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
\text{Input gate activation} \quad & i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
\text{Output gate activation} \quad & o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\text{Cell state} \quad & c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
\text{Output} \quad & h_t = o_t \circ \sigma_h(c_t)
\end{aligned} \tag{2.11}$$

In Eq. 2.11, x_t, h_t are the input and the output data vectors of the LSTM unit; W_f, W_i, W_o, b_f, b_i, b_o, are the weights and biases of the input gate, the output gate and the forget gate, respectively. σ_g is a *sigmoid* function, and σ_c and σ_h are hyperbolic tangent functions; These are the default functions of the original LSTM model and have been chosen to ensure a minimum level of generalization ability according to the LSTM cell [118]. Also, the circle represents the dot product or element-wise multiplication.

LSTM has all the abilities of RNNs but it can also preserve a long history of the

past results using self connecting loops without being forgotten. In other words, LSTM can process long data sequences and capture long-term dependencies but a major drawback is its complexity [286]. The other shortcoming with LSTM is that its training process requires a large number of samples. Also, LSTM has predominantly been used for classification purposes where the output of the model is a class denoted by a single categorical or multiple categorical variables. However, time series prediction is naturally a regression problem where the output of the model is a continuous variable. Therefore, not only are more training samples required, but also the model behavior is far from the smooth function [186] required for time series prediction purposes. The output of a time series prediction model should be smooth because its target is a continuous real variable, as opposed to the output of a classification model which its output is a categorical variable.

2.1.3 Final Summary

In this section, we reviewed the most popular parametric and non-parametric techniques used in the are of time series prediction. Each of these models can be used as a tool for implementing OSAP and MSAP methods (shown in Table 2.1). Furthermore, it was observed that each of the methods described have strengths and weaknesses which makes each of them effective for a specific type of data. This suggests that conducting an analysis on the choice of the appropriate model for the given time series can help improve the prediction performance. We study this problem in greater detail in Chapter 6.

Table 2.1: Methods: Strengths & Weaknesses

Method	Strengths	Weaknesses
ARIMA	1) Works well on linear data	1) The assumption of a rigid structure in the data, and 2) The assumption that the data can be described by a linear equation
SVR	1) SVR is robust to outliers and 2) The decision model can be updated	1) Extensive memory requirement, 2) Requires Feature Scaling, and 3) Suffers from the curse of dimensionality
NN	1) The ability to work with incomplete data, 2) Storing information on the entire network, and 3)Can be used to model non-linear data	1) Sensitive to different random weight initializations, and 2) Need for and initial hyper-parameter optimization
RNN	1) RNNs can process data of various lengths, 2) RNNs can remember past mid-term results throughout the time which is very helpful in time series prediction, 3) With increase in the size of data the model size does not need to change	1) Due to recurrent weights, the training is slow and 2) Possibility of vanishing or exploding gradients
LSTM	1) Has the ability to process long time lags, 2) In contrast to hidden Markov models, LSTM does not require tan a priori choice of a finite number of states, and 3) Can handle noise, distributed representations and continuous values	1) requires a large number of training samples, 2) Can easily overfit, 3) Requires an extensive computation power, and 4) They are highly sensitive to differing random weight initializations

2.2 Multi Step Ahead Prediction Strategies

Time series prediction has traditionally focused on predicting outcome variables using time series models incorporating a One Step Ahead Prediction strategy (OSAP). The incorporation of the time series model and the Multi Step Ahead Prediction (MSAP) strategy have been shown to have a significant impact on predictive power [11]. MSAP approaches have been used to predict wind speed [100, 261], Hydrological [18, 63, 203] and oil price time series [277, 293]. The two major MSAP implementation strategies are the direct (DIR) and the recursive (REC) approach, where both have their own specific drawbacks and strengths. A number of approaches such as Direct Multiple Outputs (DirMo), Direct Recursive (DirRec), Multiple Inputs Multiple Outputs (MIMO) have been proposed to combine these in some way to improve the the overall accuracy of predictions of MSAP [244].

In contrast to OSAP methods, where the prediction horizon (PH) consists of only one interval, MSAP is not a straightforward problem, as the additional steps of prediction add more complexities to the prediction process. Several MSAP strategies exist among which the REC and the DIR approaches (which will be explained later in this section) present two main and distinct perspectives. Other strategies suggest either an extension of one or a combination of the two, to improve the performance [45].

- **The Recursive strategy (REC)**

As mentioned previously, serial correlation between time series lags is one of the main sources of information when developing prediction models. REC repeats an OSAP method for each predicted point in the prediction horizon and will eventually use predicted lagged values as inputs for time points later in the time horizon. Generally, in REC [205], a single model performs OSAP several times during training, each time addressing one time unit (TU) in the PH. Eq. 2.12 illustrates this approach where d is the number of lags used as inputs to the OSAP method, t is time, and y is the time series.

$$\hat{y}_{t+1} = f(y_t, ..., y_{t-d}) \tag{2.12}$$

An important advantage of REC is that it only needs to incorporate a single prediction model and thus, in comparison with other MSAP strategies, REC is more efficient and thus deals with less computational complexities. This is particularly so for problems that take many inputs and require continuous long-range forecasts [101, 247]. Therefore, studies such as [101] suggest that REC is more suited for well-specified models.

However, the major disadvantage of REC is the problem of accumulation of error which occurs due the inaccuracy of the predictor. In fact, as shown in Eq. 2.12, the first step of prediction ($\hat{y}_{t+1}$) is made using actual inputs ,i.e., $y_t, ..., y_{t-d}$. But, the second step of prediction ($\hat{y}_{t+2}$) requires y_{t+1} as an input which is not available and thus, we will use the predicted value $\hat{y}_{t+1}$, instead. Likewise, the third and further steps of prediction are also made using predicted inputs. Therefore, if the prediction model is not 100% accurate, $\hat{y}_{t+1}$ (the predicted value) is not equal to y_{t+1} (the actual value) and thus, the input to the prediction model for the second and further steps of prediction will be inaccurate. This causes accumulation of error which is due to the inaccuracy of the prediction model.

- **The Direct strategy (DIR)**

DIR avoids the incorporation of lagged predicted values as inputs to future predictions by modelling each point in the PH with the same data. However, DIR uses the same inputs to make predictions for several steps and thus, lessens the impact of serial correlation which is a necessary component in time series predictions [39]. A recent study highlighted the range of research projects which are addressing these challenges [45]. Some strategies were developed to incorporate both DIR and REC methods in an attempt to overcome the shortcomings of each individual strategy. For each TU in DIR, one specific model is trained as shown in Eq. 2.13. Here, k denotes the position of the TU being modeled, and L is the size of the PH.

$$\hat{y}_{t+k} = f_k(y_t, ..., y_{t-d}); k = 1, 2, ..., L \tag{2.13}$$

However, a weakness is that DIR discards sequential correlation in a time series which negatively impacts its performance [245].

DIR was also implemented in [34] using the Least Square-Support Vector Regression (LS-SVR) as a multistep ahead prediction for daily river flow variation forecasting. The authors also implemented DIR using an NN model and compared the results with the results of the LS-SVR model.

In [102], the authors conducted a comparative analysis of DIR and REC for the performance multi-periodic time series prediction. In their research, they reported some studies suggesting that DIR can yield more accurate results. They have also reported about the studies with a belief that DIR can outperform REC depending on some factors such as the nature of the time series, the size of the prediction horizon, the input selection criteria, the prediction horizons, and the prediction period [139]. Using experimental results on six time series datasets, they suggested that DIR would yield more accurate predictions for multi-periodic time series. However, their experiments were performed on only six time series which is not sufficiently large to draw valid and generic conclusions.

- **DirRec**

In [240], a combined MSAP strategy called the DirREC strategy is presented, which combines the principles of DIR and REC. Like DIR, DirREC predicts each TU in the PH with a different model and like REC, it uses the predictions of previous TUs as additional inputs. For the time series $[y_1, ..., y_N]$, DirREC learns K models according to Eq. 2.14.

$$\hat{y}_{t+k} = f_k(y_{t+k-1}, ..., y_{t-d+1}) \tag{2.14}$$

where d is a constant showing the lags, $d \leq t \leq N - K$ and $1 \leq t \leq K$. As shown in Eq. 2.14, for each step of prediction, the model with a different number of lags is trained. For example, the first step $\hat{y}_{t+1}$ is predicted using lags $y_{t-d+1}, ..., y_t$, but the second step $\hat{y}_{t+2}$ is predicted using $y_{t-d+1}, ..., y_{t+1}$. The main drawback with DirREC is that as the number of inputs (with prediction error) increases, the complexity of the model increases correspondingly, without input selection [240].

- **MIMO**

MIMO (Multiple-Inputs Multiple-Outputs) [42] is another MSAP strategy that aims at addressing the conditional interdependence assumption made by DIR, and the accumulation of error in REC. MIMO, like DIR, considers one specific output for each TU but all outputs come from a single model. MIMO returns multiple predictions covering the entire PH by estimating one specific output for each TU as shown in Eq. 2.15.

$$[\hat{y}_{t+K}, ..., \hat{y}_{t+1}] = f_k(y_t, ..., y_{t-d}) \tag{2.15}$$

However, the main disadvantage with MIMO is that it incorporates the same model structure for each TU which effectively limits its flexibility [245]. More precisely, $\hat{y}_{t+1}$, $\hat{y}_{t+2}$, ..., and $\hat{y}_{t+k}$ are all predicted using the same inputs in one prediction model, regardless of the fact that the input y_{t-d} might have no contribution the further steps of predictions , i.e., $\hat{y}_{t+2}$, ..., and $\hat{y}_{t+k}$.

- **DirMo**

In [245], the authors combined DIR and MIMO in an attempt to take advantage of the prominent characteristics of both methods as DirMo. DirMo partitions the PH into several intervals of equal lengths (containing s TUs), each modeled using a different MIMO model. Therefore, DirMo predicts PH by training n ($n = PH/s$) MIMO models or F_k ($k = 1..n$) from time series $y_1, ..., y_n$, as illustrated in Eq. 2.16.

$$\left[\hat{y}_{N+k.s}, ..., \hat{y}_{N+(k-1).(s+1)}\right] = F_k(y_N, ..., y_{N-d}) \tag{2.16}$$

As shown in Eq. 2.16, DirMo uses the same inputs in all the MIMO models,i.e. $y_N, ..., y_{N-d}$, but the output of each MIMO model are predictions over a specific part of the PH. For example, the first MIMO model, where $k = 1$, predicts $\hat{y}_N, ..., \hat{y}_{N+s}$, or simply the first s steps; and the second MIMO model predicts the second s steps.

2.2.1 Critique for MSAP strategies

Among the existing strategies mentioned in this section, DIR and REC present the most distinctive viewpoint and other methods borrow the concept from one or both of these two approaches. In the literature, some studies such as [240] and [102] suggest that DIR is in practice a more accurate strategy since it does not suffer the accumulation of error. Many other studies such as [266] and [38] argue that theory behind DIR is based on false assumptions due to ignoring the impact of auto-correlation and suffering from the intermediate information loss. Thus, one cannot opt in favor of DIR due to observing a better accuracy over a limited number of experiments. It seems that the literature would choose REC and suggest that a limited number of successful practices cannot hide the flaw in the theory of DIR. Although, DirRec [240] was developed to improve the accuracy of both REC and DIR, studies such as [276] showed that DirRec does not always exhibit a superior performance in comparison with DIR and REC.

Some studies such as [42, 44] introduced MIMO as a better strategy than REC and DIR and claimed that MIMO has the ability to maintain the stochastic dependencies between the forecasts and is also less affected by the accumulation of error. However, studies such as [139] and [64] demonstrated that MIMO exhibits a weaker performance in comparison with DIR and REC strategies. For the most part, a MIMO model must deal with the complexity of recognizing the right inputs for each output in the same model, as not all the inputs are needed to predict each output.

Other studies such as [245] and [248] compared DIR, REC and MIMO strategies with

DirMo, and proposed that DirMo would outperform the others if optimal parameters are used. More precisely, as DirMo presents a trade-off between the flexibility of the prediction process and maintaining the stochastic dependencies among predictions, its performance depends on the choice of the controlling parameter [127]. The superiority of DirMo over DIR, REC, MIMO and DirRec was demonstrated in studies such as [6] and [5] focusing on the appropriate choice of the controlling parameter. However, recent studies also reported that the right choice of the controlling parameter of DirMo may only keep its performance high in short horizons and may not guarantee the performance over long horizons [5]. In [27], the authors attempted to convert DirMo into a particle swarm optimization (PSO) problem and use the meta-heuristic abilities of PSO to find the optimal parameters for DirMo. However, [294] argued that the computational requirements of the PSO solution for optimizing DirMo parameters was excessively high, and thus PSO-DRIMO is not recommended.

An overview critique of existing MSAP strategies is provided in Table 2.2 to summarize this section on MSAP strategies.

Table 2.2: MSAP Strategies: Strengths & Weaknesses

Strategy	Strengths	Weaknesses
Rec	Accounts for serial correlation	Accumulation of error
Dir	Uses actual inputs to make predictions	Is computationally complex and suffer from the intermediate information loss (ignores serial correlation)
DirRec	Was developed to preserve the stochastic dependencies between forecasts	Growing complexity in long horizons, Inaccuracy without input selection
MIMO	Attempts to preserve stochastic dependencies between the forecasts	The complexity of input selection
DirMo	Attempts to preserve stochastic dependencies between the forecasts	Is highly sensitive to the choice of the controlling parameter

As can be seen in Table 2.2, only REC does not suffer from the intermediate information loss problem. In fact, REC in theory is able to make significant forecasts, and the problem of the accumulation of error only occurs as a result of the inaccuracies of the predictor. One of our goals is to develop a new MSAP approach which adopts the principles of REC and uses the forecasts at multiple resolutions to improve the accuracy of MSAP.

2.3 Generating Synthetic Time Series

Performance assessment is the ultimate phase of the process when a new time series method is developed. Practitioners often test their proposed methods on a large volume of data to ensure the assessment quality. A new algorithm is tested over various types of data (i.e, data with different features, shapes, distributions, lengths from different domains and sources) to also identify its strengths and weaknesses. Therefore, diversity is highly required for a robust assessment. There are numerous competition and open source datasets that have been made available to the research community, such as BCI [110] and Kaggle [246]. These datasets have allowed researchers to test their methods more extensively. However, diversity is one the most important features and is not provided in these datasets.

In the absence of appropriate datasets, researchers had to use artificial datasets and incorporate surrogate or simulated/synthetic data to create a dataset that represents the required properties.

Surrogate Data.

Data generation has been widely used in time series analysis through the use of surrogate data analysis [236], synthetic data generation [231] or simulated data [262]. The surrogate method was initially introduced to differentiate between linear and nonlinear processes [168]. The surrogate data are created to have the same statistical attributes (such as the mean and variance) as the original data. However, they are typically generated as a linear stochastic process [267].

Surrogate data analysis can be used as a means of estimating the impact of the scale of a characteristic in a time series, through the comparison of the given time series with surrogate series [236]. This can be demonstrated by estimating for example, the impact of non-linearity in a time series in comparison with a series generated from a linear models and thus, allows researchers to replicate statistical features such as auto-correlation. Autocorrelation is a class of serial dependence and defined as the correlation between successive data points of a time series [210].

Synthetic Data.

Many studies use synthetic data to practice performance testing. Studies such as [134], [190] and [130] argue that the available large datasets are relatively homogeneous which limits their applicability for performance assessment purposes of general time series methods. Smith-Miles in [237] asserts that the performance of time series prediction methods relies on the *diversity* of the training data in order that diversity allows the performance assessment to be generalized to a broad range of unseen cases. Diversity is also an essential property for a time series benchmark as a robust evaluation of time series methods requires the algorithm be tested over various types of data [134]. The term "Homogeneous" points out a class of time series which show the same types of features but may only be different in local fluctuations or temporal patterns. Diversity was used to address time series which are different in any aspects such as distribution, the source process, low frequency movements, entropy, complexity, memory.

Many studies generated synthetic time series to enhance their performance evaluation. The majority of practices in time series generation typically assume a linear structure such as the ARMA family of models [252]. These models establish fundamental statistical consistency, by means of reproducing the mean, variance and autocorrelations of lags of the parent historical data [231]. However, many real-world time series show substantially more complex statistical properties; for example, time series with skewness rather than Gaussian distributions, or those characterized by statistical inter-dependencies [251].

In [231], a Markov chain model was used to generate synthetic data for a wind speed time series analysis. Characteristics such as mean, standard deviation and frequency distribution are predominantly used as assessment metrics. They also evaluated autocorrelation and power spectral density to determine the persistence structure of the series.

In [260], the authors presented a method that incorporates maximum entropy bootstrapping to generate ensembles to create a large number of replicates from the given time series data. Bootstrapping is a statistical process that allows for the estimation of the statistics on a population by (usually random) resampling the given dataset with replacement. The authors used a seven step algorithm to create the replicas and their algorithm needs to satisfy the ergodic theorem where the mean of the original sample is preserved unchanged in the samples being generated. The authors demonstrated that their method can retain the basic shape and the time dependence structure of the time series' autocorrelation function. However, obviously, this method only focuses on low frequency approximation of the signal and discards memory characteristics that are present in temporal fluctuations.

Niu and Sivakumar [194] used the Morlet wavelet transform and developed a scale-controlled approach for generating synthetic data for the daily streamflow series of the Pearl River basin in China. In this work, the Morlet wavelet transform first decomposes the original time series and then the synthetic data are created by performing the reconstruction on a random permutation of the categorized wavelet coefficients.

In [84], a stochastic approach was presented to simulate long range persistence of hydro-meteorological time series at multiple scales. The authors use a linear stochastic model to generate synthetic data that replicates the Hurst-Kolmogorov characteristics of the original process. The authors also claim that for their dataset, their method could replicate both the stochastic (such as the mean and the variance) and the Hurst-Kolmogorov as well as the seasonality properties of the original time series, which is not provided by the existing methods at the time. However, this method attempts to *replicate* temporal dynamics to create similar series.

In [19], the authors tried to use white noise to generate new patterns. This work was mainly conducted on providing insight about the discriminatory attributes of the data and presented an algorithm that exploits five different features to generate randomized time series data. The presented algorithm in this work makes use of the Hierarchical Collective of Transform based Ensembles (HIVE-COTE) approach to generate a set of time series with the same representation but dissimilar shapes. This work was originally conducted to compare the performances of time series classification methods on the data for variant representations. However, in this work, the authors focused on preserving the representation of of the given time series and the implementation of diversity was only limited to local random alterations of the shape of the patterns.

More recently, a machine learning technique known as Generative Adversarial Networks (GAN) [62] was used to generate similar datasets (almost identical in the low frequency shape but different in high frequency fluctuations). GANs were originally introduced as an approach that facilitates generative modeling via deep learning. Despite adversarial training, GANs can implicitly learn the intrinsic dynamics of time series and provide the ability to generate scenarios that share many similarities to the original time series.

The GANs' training process forces the output of the network to follow the distribution of the given input. Most studies on GANs have focused on image generation, but there is limited work on time series data. However, [185] did use GANs to generate continuous sequential data representing classical music pieces. A similar attempt was also made in [80], where GANs were used to reproduce musical symbolic sequences.

In [86], the authors presented a Recursive GAN to create realistic looking medical time series with the purpose of improving supervised learning (providing an abundance of training samples). In [107], GANs were used to generate biosignals such as EEG and ECG signals. Physicians usually use the patterns observed in these signals to diagnose diseases and disorders, and make decisions about the potential treatments. This work tried to generate synthetic biosignals to provide enough training

samples for the machine learning tools employed to detect disorders based.

In [70], the authors presented an approach which makes use of hidden Markov models (HMMs) and regression models for generating realistic synthetic behavior-based sensor data to enhance the performance of machine learning approaches when applied to health-care time series. This approach which in called SynSys was used to generate activity sequence time series with the use of random processes. However, since this work intends to create realistic looking time series, its output time series need to be similar to the input series and thus cannot provide diversity.

In [204], the authors used GANs to generate a large amount of financial time series in order to mitigate overfitting in machine learning time series models.

Past studies have also tried to use GANs to create diverse artificial data [292]. However, GANs were originally developed to create diverse data samples which have similar distributional characteristics as the original data. Thus, the application of GANs are limited to artificial training sample generation which can be used to enhance supervised learning techniques.

More complex approaches have used deep learning for synthetic data generation, such as [10]. In this work, the authors proposed a deep learning architecture which incorporates a stack of multiple Long-Short-Term-Memory (LSTM) networks and a Mixture Density Network (MDN) for Synthetic Sensor Data Generation. This work attempted to develop a model that reproduces similar sequences of data which preserve specific statistical properties.

Table 2.3: Existing Synthetic Data Methods & Diversity

Ref.	**Year**	**Approach**	**Goal**	**Diversity Focus**
[231]	2005	Markov chain	Provide numerous samples for training purposes	No
[260]	2009	Maximum Entropy bootstrapping	Preserves the time dependence structure of ACF	No
[194]	2013	Morelet wavelet transform	Generate daily streamflow time series	No
[84]	2014	Catalia	Preserve the stochastic and the Hurst-Kolmogorov properties	No
[185]	2016	GANs	Create similar musical pieces	No
[19]	2017	HIVE-COTE	Create randomized replicates of a time series that have the same representations	No
[86]	2017	Recursive GANs	Provide similar looking time series	Yes / To generate an abundance of samples
[10]	2017	Stack of LSTM and MDN	Synthetic Sensor Data Generation	No
[80]	2018	GANs	Produce similar musical sequences	No
[62]	2018	GANs	Generate similar time series	Yes / Provide more training data
[107]	2018	GANs	Generate synthetic bio signals	No
[70]	2019	SynSys via HMMs and regression models	Create realistic looking activity sequence time series	No
[204]	2020	GANs	Generate a lot of time series	Yes / Mitigate overfitting in machine learning problems

Our review of the construction of synthetic time series confirms the views of [169] that the lack of a diverse benchmarking dataset for performance evaluation is a big challenge in the area of time series area. Table 2.3 attempts to capture the lack of focus on this issue of diversity across the various research efforts we analyzed. The existing synthetic time series generation methods all focus on providing similar time series of the same general behavior, training samples for machine learning techniques or an abundance of samples. Thus, this motivates the need for a new approach to generating synthetic time series.

2.4 Approaches Using a Meta-Learning

As the field of time series analysis has expanded, the number of available methods/approaches has also increased and thus, has induced a greater level of complexity when choosing a method/approach. Identifying an appropriate prediction method for a time series dataset has evolved into a new area of research known as Meta-Learning [56]. The concept behind Meta-Learning is to analyze a dataset for a range of specific characteristics, which can then be compared to a metabase of pre-learned methods, [14, 15, 155, 262].

A comprehensive review of the literature shows that the general trend in method choice follows the approaches used by predecessors who have had similar data types [154, 281]. For instance, in some domains (such as the wind speed prediction), the time series data are characterized by a phenomenon known as Chaos [99].

Another approach to prediction method selection was suggested in [179] which was based on the *goodness* of the prediction performance. In simple terms, the method to show the least prediction error is recommended to the user. In [37], a method selection approach over exponential smoothing methods was proposed. This work compared the traditional validation-based model selection approach by the information criterion and showed that the information criterion approach exhibits a better basis for model selection. The research conducted in [222] was one of the earliest

studies suggesting that the accuracy of prediction methods changes in relation to the properties of the time series. Studies on the prediction performance like the M-Competition have somehow confirmed this statement [171].

An expert system was suggested in [149] that extracts some characteristics from the time series and recommends the appropriate method using a pre-trained knowledge based system. The research conducted in [65] was the first of the kind that used NNs as the method selector. The input of the NN was a set of features and candidate predictions made by prediction methods and the output was the final forecast. Thus, basically, this method uses an NN to choose which candidate forecast is the best and returns the final forecast directly. However, most approaches recommend the candidate method and this work returns the candidate forecast.

Arinze in [14] suggested a rule induction based approach for prediction method selection. This work trained a decision tree (the ID3 technique), which is a classification technique, based on a set of time series features to recommend the appropriate prediction method. Based on the trained tree, a set of rules were then induced representing the conditions in the feature space through which the prediction methods are recommended. In [229], the authors used 26 features to train a discriminant analysis (DA) meta-learner in order to recommend the appropriate prediction among a few statistical approaches.

Armstrong, in [15], suggested another approach that uses expert knowledge as well as time series features to recommend the appropriate prediction methods. This work also suggests a set of criteria for selecting the appropriate prediction method. In [212], a prediction method selection approach was presented that works based on ANOVA and Duncan multiple range tests on time series data. This method selection approach were only applied to ARIMA, regression and a decomposition based method.

More recently, prediction method selection was addressed as a meta-learning problem where the method selection interface is obtained via the use of machine learning techniques.

In [262], a rule induction approach was presented for prediction method selection. In this work, the Self-Organizing Map (SOM) clustering and Decision Trees (DTs) classification techniques were applied to a set of characteristics including measures for chaos, self-similarity and traditional statistics like trend, seasonality and kurtosis to extract a set of interface rules for prediction method selection.

In [156], the authors built a large pool of meta-features and tried to relate the model performance to these features using a number of approaches as the meta-learner, including: NN, Decision tree, SVM, zoomed ranking of the best and zoomed ranking of the combination. The Zoomed-Ranking method is an approach to ranking multiple candidate models for a given time series prediction problem; The zoomed-ranking approach ranks models based on their accuracy and execution times on a set of data which are similar to the given time series. Information The work [215] suggests a prediction method selection approach that recommends the appropriate method based on out-of-sample rolling horizon weighted errors, which is an extension of the classic selection criteria that uses the minimum one-step out-of-sample error for performance evaluation.

In [269], the authors presented a prediction method selection approach which recommends the appropriate prediction method based on its previous performance on similar datasets, and requires a database that keeps the historical records of the predictors' performances. The similarity of time series' datasets is measured based on a set of time series characteristics. In this work, Principal Component Analysis (PCA) was used to reduce the dimensionality of the data.

Prediction method selection was studied in [89] for chaotic time series. This approach works based on meta-learning and the Self Organizing Map (SOM) and Monte Carlo Cross Validation (MCCV). Monte Carlo Cross Validation generates several random splits of the dataset into training and validation sets. For each split, the model is trained by the training data, and the performance is evaluated using the validation set. The final result is reported as the average of the error over the splits of data.

SOM provides a topology preserving mapping from the high dimensional space to map units that preserves the relative distance between the points. Points that are near each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as a cluster analyzing tool of high-dimensional data [140].

In [147], the authors studied the predictive accuracy of using different feature sets for a neural network meta-learner which recommends the appropriate method from a set of four statistical forecasting models, This work makes use of error based features (landmarkers) and statistical tests as time series meta-feature.

In [238], the authors presented an approach referred to as the self-learning (method selection) approach that conducts cluster analysis to recommend the most appropriate prediction method.

When reviewing all of the papers above, we determined that an effective comparison should focus to the following factors:

1. Prediction models
2. Features
3. Selection criteria
4. The incorporation of hyper-parameter selection
5. The provision of experimental results
6. The type of meta-learner (the method selection approach)

Prediction models. Prediction models are the candidate models among which the meta-learner chooses its recommendation. These models can be classified into two types of models: stochastic models and machine learning models.
There are different types of **features** used in the related works, which can be categorized into four general categories including:

- Statistical features such as variance and Skewness

- Advanced features such as frequency domain features, Hurst and DFA which are explained in details in in Chapter 5, section 5.3.1
- Judgemental features such as strong trend and non-linearity (used in [262])

Selection criteria. The Selection criteria is the metric based on which a candidate model is chosen among the available models. The most popular criteria is error which indicates the out-of-sample accuracy of a candidate models over a set of datasets. AIC, Variance, selection rules and Monte Carlo Cross Validation (MCCV) are the other criteria used in the literature of this research.

Hyper-parameter selection. Hyper-parameter selection is an important phase of the modeling process, specially when using machine learning techniques, and the model's performance is not reliable without an appropriate hyper-parameter selection. Therefore, when machine learning techniques are used as candidate models, hyper-parameter optimization should be performed before applying the meta-learning approach.

Experimental results. Many studies support the evaluations by providing Experimental results, which is can demonstrate the ability of the proposed method in practice.

Among the related works, a few studies failed to support their research via the provision of experimental results.

Meta-learner. Finally, the Meta-learner (the method selection approach) is the core of a meta-learning mechanism, where the decision making algorithm is implemented. For instance, [262] used DT as the meta-learner, while [238] used a cluster analysis approach to implement their meta-learning approach.

Using the mentioned factors, a summary of the existing meta-learning approaches for time series prediction is provided in Table 2.4. In Table 2.4 - column *Selection criteria*, Err, Var, RI and MCCV are the short terms for Error, Variance, Rule Induction and Monte Carlo Cross Validation, respectively. Also, in column *Feature vector size*, S, Med, L and VL are the short terms for Small, Medium, Large and Very Large, respectively.

Table 2.4: Comparison between the existing meta-learning approaches

Works	Year	Stochastic models	Machine learning models	Statistical features	Advanced features	Judgmental features	Feature vector size	Selection criteria	MSAP	Hyper-parameter selection	Experimental results	Meta-learner
[37]	2006	✓	✗	✗	✗	✗	✗	AIC	✗	✗	✓	✗
[149]	1988	✓	✗	✓	✓	✓	L	✗	✗	✗	✗	ES
[65]	1994	✓	✗	✓	✗	✓	L	Err	✗	✓	✓	NN
[14]	1994	✓	✗	✓	✗	✓	M	RI	✗	✗	✓	ID3
[229]	1997	✓	✗	✓	✗	✓	L	Err	✗	✗	✓	DLA
[15]	2001	✓	✗	✓	✗	✓	L	Err	✗	✗	✗	Rules
[4]	2001	✗	✗	✓	✓	✓	L	Err	✗	✗	✓	RBF
[217]	2004	✓	NN	✓	✓	✗	L	Err	✗	✗	✓	NOEMON
[262]	2009	✓	NN	✓	✓	✓	L	Err	✗	✗	✓	DT/SOM
[212]	2010	✓	✗	✓	✗	✗	S	Var	✗	✗	✓	ANOVA
[156]	2010	✓	✓	✓	✓	✗	VL	Err	✗	✗	✓	DT
[215]	2011	✓	✗	✗	✗	✗	✗	Err	✗	✗	✓	ES
[269]	2013	✓	✗	✓	✗	✓	S	Err	✗	✗	✓	DR
[89]	2014	✗	✓	✓	✗	✗	S	MCCV	✗	✗	✓	SOM
[147]	2016	✓	✗	✓	✓	✓	VL	Err	✗	✗	✓	NN
[238]	2016	✓	NN	✓	✓	✗	M	Err	✗	✗	✓	Cluster-Anal
[213]	2018	✓	✗	✗	✗	✓	M	Err	✗	✗	✓	Judgements

By analyzing the results in Table 2.4, it can be seen that most of the related works use stochastic models in their analyses. Among these works, only [156], [217], [262], [89] and [238] have incorporated machine learning techniques to implement their prediction models. Studies including [217], [262] and [89] only implemented NN models in their studies. Studies such as [156] and [238] were the only studies that

made use of the state of the art techniques, such as RNN and LSTM, in their analysis. RNN and LSTM have recently received considerable attention due to their abilities in processing time series data, and thus play an important role in current time series prediction studies. However, special considerations are needed to be taken into account when using these methods as they are highly sensitive to the choice of their hyper parameters. Despite the the use of RNN and LSTM as candidate models in [156] and [238], both of these studies failed to provide a valid evaluation in their works as they did not pay attention to the problem of the hyper-parameters selection. Hyper-parameters selection is an important concern when implementing machine learning models because the capacity and the performance of ML techniques are highly dependent upon the optimal choice of hyper-parameters. We also see that none of the existing works represent the ability to recommend an appropriate model for MSAP problems.

This final section of our literature review articulates the need for the last part of our research contribution: a new meta-learning framework that uses a bootstrapping approach to ensure the proper incorporation of RNN and LSTM in the selection process. This should also suggest the appropriate model for MSAP problems where others provided only recommendations for OSAP problems.

2.5 Summary

In this Chapter, we reviewed the literature of time series prediction in the areas of Multi-step Ahead Prediction, Synthetic time series generation and Meta-learning (for prediction method selection). We showed that several MSAP strategies have been proposed, such as REC, DIR, MIMO, DirREC and DRIMO. However, the problem of the accumulation of error in REC based strategies and the intermediate information loss in DIR (as well as MIMO, DirREC and DirMo) have kept the area open to development of new MSAP strategies. We also learnt that the existing synthetic time series generation approaches failed to provide datasets sufficiently robust to assess time series models. Finally, a review of meta-learning methods

revealed that the existing approaches fail to focus on the problem of hyper-parameter selection in machine learning models and also lack the ability to recommend MSAP models.

Chapter 3

Solution strategy

In Chapter 2, we reviewed the state of the art in terms of the research questions and goals presented in Chapter 1. This Chapter presents an overview methodology and research plan for how the main contributions of this work are delivered. In §3.1, we describe our overall approach and how we divide our research into a number of logical steps. We then proceed to discuss our initial research into multi-step ahead predictive models in §3.2, before motivating the requirement for synthetic time series creation in §3.3, together with a description of our approach. In §3.4, we outline the final step, which is the application of meta-learning to our overall research methodology. In each case, we will articulate the scope and limitations to this research before a summary is presented in §3.5.

3.1 Outlining Our Approach

The overall approach to this research can be outlined using figure 3.1. There are 3 clear steps, briefly outlined in this Chapter and described in detail in Chapters 4, 5 and 6. Each step has a validation component and at the conclusion, a final validation takes place over all 3 steps to this research.

In figure 3.1, the first step represents our attempt to improve on predictive time

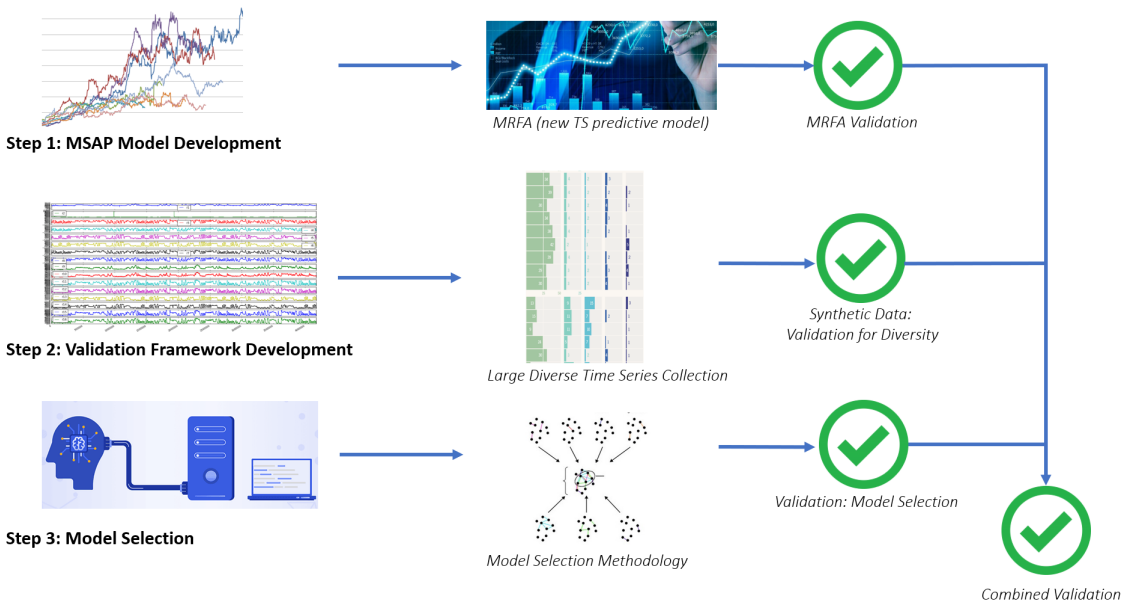


Figure 3.1: Outline Methodology

series models in the particular area of multi-step ahead prediction (MSAP). As will be shown, this step results in a new MSAP model and our evaluation of this work provided a clear requirement that we had not considered at the outset of this research. In brief, the major finding from this first step was the realisation that a far larger set of time series was required to deliver the robust evaluation that all new time series models require.

As a result, step 2 involved the creation of a large data collection together with a validation framework to ensure that of the dataset was properly representative in terms of the spread of time series characteristics. A total number of $50k$ time series were created during the experiments, each has a length between 400 and 45,000 samples. The validation for step 2 determines and measures these characteristics as they occur in the generated time series.

Initially, the creation and validation of synthetic time series was regarded solely as a means to deliver robust validation of new MSAP models. However, this step also highlighted the task facing time series researchers where the suitability of models in the face of large numbers of time series, presents a challenge in itself. How does one select an appropriate model when large numbers of time series are required

for testing? Thus, the logical step for us was to devise a meta-learning method to support time series researchers in the selection of the appropriate model, exploiting the makeup (characteristics) of the particular time series.

The final step is a validation which takes all three components presented in this dissertation and deliver an overall evaluation with insights from the entire breadth of our research.

3.2 Developing New MSAP Models

In §2.2, we reviewed the literature of MSAP approaches and learnt that among the existing methods, only REC has the proper theoretical basis to account for the serial correlation (the autocorrelation structure) between time series lags. The problem with REC is that as the prediction horizon extends, errors start to accumulate. The review of the literature demonstrates that a large portion of prior research relies on the auto-correlation structure of the time series to implement their prediction models. Other than REC, the existing approaches suffer from the intermediate information loss problem, which in practice is equivalent to making prediction based on incomplete data (lag 1 is lacking, at least). Therefore, we focus solely on REC and try to use the principles of REC to develop a more effective MSAP strategy.

Among the machine learning techniques reviewed in §2.1.2, and by incorporating recurrent feedback loops, RNN and LSTM have shown their ability to process time series data. However, LSTM requires a large number of training samples which is not always available. Also, LSTM has predominantly been used to solve classification problems, where the output of the model is an finite number of possible choices. On the contrary, time series prediction is fundamentally a regression problem. This motivates the requirement for a new approach to MSAP time series prediction and this lies at the core of our research.

Our new approach to the MSAP predictive strategies is known as Multi-Resolution Forecast Aggregation (MRFA), and incorporates an additional concept known as

the Resolution of Impact (ROI). In Chapter 4, we present a new strategy for MSAP problems which has the ability to alleviate error accumulation by retaining a *far higher proportion* of actual historical data for *each* long term forecast. We will present a novel MSAP strategy which uses machine learning models as a tool to achieve our goal. We do not attempt to improve recurrent neural networks (RNN), rather we use RNNs as a tool to implement our MSAP strategy. Additionally, this is not an attempt to improve MSAP across *all classes* of time series data, but to seek some improvement in the recursive strategy (REC) by incorporating more accurate *prior* long term forecasts; A prior long term forecast is a prediction which was previously made over multiple intervals.

Scope and Limitations. In terms of MSAP models, we can now state the scope of our research. We limit our focus to univariate time series data. Since the proposed approach is an MSAP strategy, it is independent of the modeling technique and thus, any type of regression models such as NN, RNN and SVR can be used to implement our strategy. We should also articulate the limitations to this research component. We were unable to implement our strategy using LSTM due to the lack of computational resources. Secondly, we only compare our strategy with REC, since other approaches fail to retain the auto-correlation structure of the data.

3.3 Building a Robust Validation Framework

After developing the new Multi-Resolution Forecast Aggregation (MRFA) approach, our research required a performance evaluation strategy to identify its strengths and weaknesses. Having studied the literature, we realized that a robust performance evaluation strategy for MRFA requires an appropriately large dataset. However, as many other researchers have found, this dataset did not exist. Note that, when "dataset" is used in a singular form, it means a collection of datasets.

As discussed in Chapter 2, researchers will ideally test their algorithms on a large number of time series to establish their performance and capabilities. However,

a large evaluation dataset is mainly used to ensure the algorithm is tested over a large percentage of the feature space. The feature space is a multi-dimensional space where each dimension represents a separate feature. Therefore, an appropriate benchmark is one that covers a higher percentage of that metric space. We refer to this requirement as *diversity* and our assertion is that an appropriate benchmark for time series methods assessment should contain a *diverse* set of time series, or should exhibit a high level of *diversity*. Unfortunately, such a dataset is not available for public use and the literature has not provided a standard source for such datasets. The literature shows that when the appropriate dataset is lacking, practitioners create their own datasets to resembles the required characteristics but the literature also shows that in these cases, little attention was paid to diversity. As shown in Chapter 2 using Table 2.3, most of the existing synthetic time series generation methods (with any focus on diversity) have focused on providing training datasets for machine learning techniques.

Fundamentally, training datasets and performance evaluation datasets have different data characteristics. A training dataset cannot have a high level of diversity, since a training process should construct a model to solve a *specific* problem that is equivalent to a set of domain-specific features. However, diversity requires going *beyond* a specific domain.

A training set is a large subset of the complete dataset and is used to train an algorithm. A performance dataset is a large dataset to test the algorithm against various aspects. In Chapter 5, we present a framework and methodology for the generation and validation of synthetic time series that shows a high level of diversity.

Scope and Limitations.

We can now articulate the scope for the second major research undertaking of this dissertation. As stationarity is a prerequisite for time series prediction algorithms, we focus on a subset of stationarizable processes to generate our series. Furthermore, we limit our focus to component level interactions of time series data, which is a typical approach in time series prediction applications. Therefore, the first as-

sumption is the data can be decomposed into a set of components (trend, seasonality and irregularity) and as such, this decomposition has a real-world implications. We simulate additive and multiplicative interactions when combining time series components. As different time series data can be constructed by specific functions types, we will present a generic formula for time series construction. As a result, many systems may fall out of the scope of this study, such as chaotic time series which have dynamic structures.

We now highlight some limitations to our study. Seasonality is a repetitive pattern and in general, can take any shape or form. However, without any loss of generality, we consider only four types of seasonality to create our time series: sinusoid, impulsive, rectangular and step-wise. We use only difference-stationary processes to simulate the irregular component. Since measuring cyclicality is a very difficult problem, we *combine* Trend and Cyclicality and use the term *trend-cycle* component to represent this combination. Many time series attributes are uncontrollable and cannot be jointly injected into time series data in a manual manner, such as conditional heteroscedasticity and chaos. With uncontrollable attributes, we mean that there is no known function to produce attributes $A_1, A_2, ...$ with pre-specified arbitrary values $A_1 = 1, A_2 = .36, ...$ into a single time series. For this reason, there are various time series attributes that are not considered in our time series generation approach. Trend, Cyclicality, Seasonality and Irregularity are shown in Fig. 3.2.

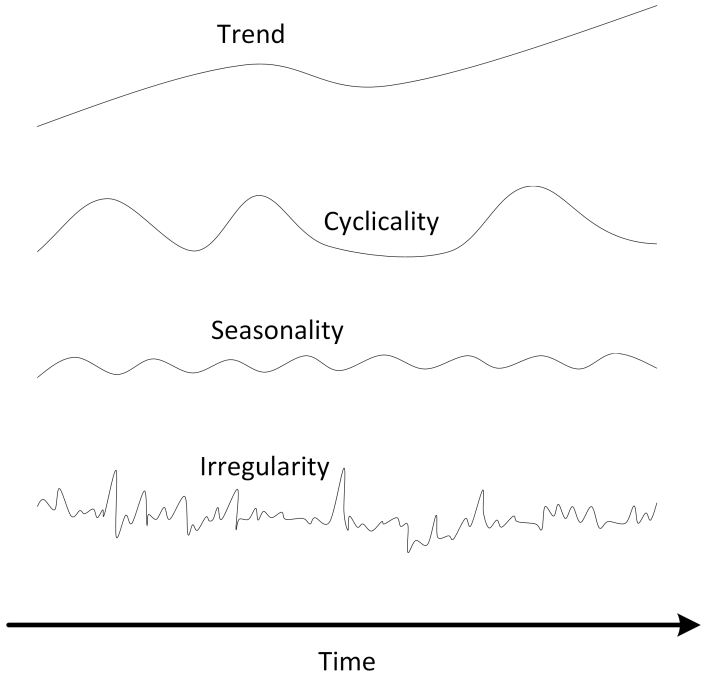


Figure 3.2: Time series components

3.4 Meta-Learning

Once it can be shown that a very high number of sufficiently diverse time series can be generated, this research will be in a position to evaluate MRFA and compare its performance with other methods. The experimental results in Chapter 4 will suggest that MRFA does not outperform other methods over the *entire* dataset. A review of the literature showed that we cannot expect an algorithm to outperform against all other models and the literature refers to this finding as the *No free lunch* theorem.

Our analysis demonstrated that every time series has a set of predictive requirements that only a narrow set of methods can provide the appropriate predictive abilities. In other words, each time series has unique properties which each prediction model performs well on a specific set of properties. Therefore, to find the best prediction method for the given time series, one should pay attention to the time series *features*.

In Chapter 2, we showed that the literature refers to this problem as the method selection problem. The method selection community adopted machine learning as a powerful tool and converted the method selection problem into a learning problem and referred to it as *meta-learning*. In meta-learning, a machine learning model is trained to recommend an appropriate prediction method at the output based on a set of time series features at the input. The literature also identified two significant weaknesses in the existing meta-learning approaches. Firstly, most of the past studies used only stochastic models and did not study the use of machine learning techniques as their candidate prediction models. Second, past studies on method selection failed to identify the essential role of hyper-parameters in the performance of machine learning models. Hyper-parameters determine the capacity of machine learning models and thus, the model's performance cannot be measured accurately if the hyper-parameters are not selected appropriately.

In other words, sub-optimal hyper-parameters yield a sub-optimal model which is crucial because the meta-learner compares the performances of the candidate models to recommend the preferred method. Thus, comparing sub-optimal models will not lead to any great improvement in performance. In Chapter 6, we introduce a meta-learner that includes RNN and LSTM as candidate prediction models and also deals with the problem of hyper-parameter selection using a bootstrapping mechanism.

We used REC and MRFA as the MSAP strategies and implemented them using five candidate models including ARIMA, NN, RNN, SVR and LSTM. The Limitations to this part of our research stem from the considerable computational requirements necessary to fully validate our theory using a far wider set of machine learning techniques. For this reason, it was necessary to stick to a smaller set of models.

3.5 Summary

We began our research with the purpose of better understanding the issues faced by practitioners when solving time series prediction problems. In particular, we carried

Table 3.1: Case study: Factors that are important when choosing a prediction model

Publication	The method	Relevance to Agri-sector	Relevance to prices prediction	Low frequency	Time series	Non-stationarity	Potential seasonality	Multi-variate time series	Robust to noise or volatility	Conditional heteroscedasticity	Random Walk	Publisher quality (High/Low)
[166], 2002	ARIMA	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	H
[257], 2013	ARIMA	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	H
[36], 2005	ARIMAX	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗	H
[275], 2017	ARIMA	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	L
[264], 2011	ARIMA	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	H
[197], 2006	ARIMA	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	L
[103], 2010	ARIMA	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	H
[243], 2011	ARIMA	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	H
[165], 2000	ARIMA	✓	✗	✗	✓	✓	✓	✗	✗	✗	✗	L
[192], 2013	ARIMA	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	L
[7], 2011	SARIMA	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	H
[209], 2013	ARIMAX	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	H
[67], 2007	ARIMA-W	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	H
[131], 2013	ARFIMA	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓	H
[13], 2011	GARCH	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	H
[218], 2003	GARCH	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	H
[268], 2004	GARCH	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓	H
[298], 2007	NN	✓	✓	✓	✓	✓	✓	✗	-	✗	✓	H
[239], 1992	NN	✓	✓	✓	✓	-	✗	✗	✓	✗	✓	H
[284], 2009	NN	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	H
[26], 2006	NN	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	H
[180], 2012	NN	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	H
[283], 2013	NN	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	H
[126], 2014	NN	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	H
[71], 2005	NN	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	H
[132], 2005	NN	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	H
[106], 2007	NN	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	H
[164], 2010	NN	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	H
[230], 2007	NN	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	H
[158], 2013	NN	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	H
[120], 2005	NN-W	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	H
[111], 2013	RBF	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	H
[296], 2012	RBF, NN	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	H
[298], 2007	ARIMA-NN	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	H
[143], 2009	ARIMA-NN	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	H
[184], 2017	ARIMA-NN	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	H
[282], 2016	ARIMA-RBF	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	H
[114], 2016	Deep NN	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	H

out a case study on the Agri sector as a strategically pivotal sector of the Irish economy. Table 3.1 shows a comparison of the related works taking into account the factors determined to be important when choosing a particular method for a range of time series prediction applications.

Table 3.1 illustrates the connection between the identified factors and the model type and can be summarized as follows:

- Conditional Heteroscedasticity cannot be easily handled by non-parametric methods without incorporating GARCH models, as none of the research using NN models could handle Conditional Heteroscedasticity. Studying ARCH and GRACH is out of the scope of this research, please see [85].
- 5% (1 out of 17) of the parametric based methods and 16% (4 out of 25) of the non-parameteric based methods showed robustness against noise or unknown volatility. This suggests that non-parametric (machine learning) models have shown better ability to deal with noise or unknown volatility.
- Machine learning methods can sometimes deal with seasonality without the use of seasonal differencing as 16% (6 out of 25) of the machine learning based methods did not require the data to de-seasonalized.

In this case study, we found that in comparison to parametric methods, machine learning techniques have received a lot more attention recently. However, ARIMA has remained consistently popular, as researchers have used it in a wide variety of applications. We also found that ARIMA has also been combined with machine learning techniques such as NNs and Radial Basis Function (RBF) networks in order to improve prediction performance. This case study highlights the potential advantage of using time series metrics as a means of directing researchers to appropriate models. However, one of the major weaknesses in current time series analysis is the lack of available disparate time series datasets to truly test the appropriateness of new MSAP strategies. Having a framework to benchmark algorithms would be invaluable to researchers, and would allow for the creation of an automated model

selection recommender system.

Chapter 4

Multi-Resolution Forecast Aggregation

As shown in Chapter 2, there are two distinctive perspectives when solving MSAP problems, known as the recursive or REC strategy and the direct or DIR strategy. Before presenting any new MSAP model, it is important to understand how these strategies are impacted by outstanding research issues. In §4.1, we discuss the three challenges in MSAP predictive modelling that we address as part of our research. In §4.2, we describe how we resolve these three challenges in our MSAP approach as they influenced our decision making and design. In §4.3, we present our multi-resolution forecasting strategy called MRFA as a new form of MSAP modelling, and deliver a comparative analysis to validate and understand the impact of MRFA in §4.4. Finally, in §4.5, we summarize our research to this point and outline the next step.

4.1 Introduction

A wide variety of phenomena are characterized by time dependent dynamics that can be analyzed using time series analysis methods. Various time series analysis

techniques have been presented in the literature, each addressing certain aspects of the data. In time series analysis, forecasting is the process of determining what may happen at a later time point. Forecasting methodologies have traditionally fallen under two broad groups; one step ahead prediction (OSAP) which seeks to predict the next time point for a given dataset, and multi-step-ahead-prediction (MSAP) which seeks to predict multiple time points ahead.

Time series prediction has applications in various fields, however, the length of the prediction window varies depending on the requirements of the problem. In general, one can classify predictions into either a short-run (single prediction) or long-run (prediction of multiple responses) time horizon. Extending an analysis from a short run, or OSAP, to a long run response is known as MSAP and the period applied to the MSAP is known as the Prediction Horizon (PH). While MSAP and OSAP problems differ with regard to length of the prediction horizon, challenges are still prevalent in both problem sets. In this chapter, we focus on three particular challenges when solving an MSAP problem: serial correlation, uncertainty, and long term memory.

Motivation.

As reviewed in Chapter 2, existing MSAP approaches have been developed under the influence of two distinctive viewpoints presented by the recursive (REC) or the direct (DIR) strategies. However, as described in Chapter 2 section 2.2, DIR suffers from intermediate information loss as a result of discarding serial correlation, while REC suffers from error accumulation due to the inaccuracy of the predictor. Existing solutions tend to address a trade-off between these two perspectives [21, 203, 244]. However, except for REC, other strategies do not pay attention to the role of serial correlation and thus, only REC can provide a valid theoretical basis for MSAP. Our approach is to develop a new MSAP strategy known as Multi-Resolution Forecast Aggregation (MRFA), which incorporates an additional concept known as the *Resolution of Impact* (ROI). MRFA is shown to have favourable predictive capabilities in comparison to a number of state of the art methods.

REC predominantly relies on serial correlation and DIR predominantly relies on the use of the actual data to predict the entire PH (which is a type of long term memory). The nature of time series data dictates that the choice between the two forms is not a binary choice problem. Researchers have attempted to create hybrid forms of both strategies, such as the DirRec, MIMO and DIRMO strategies. However, these strategies are either inflexible (DirRec) or overly complex (MIMO and DirMo). Capturing the long term memory and serial correlated dynamics of a series requires the strategy to adapt to the individual time series. In this research, we propose a more sophisticated approach that uses the principles of REC and DIR in an *adaptive* way.

Contribution. In this Chapter, a novel MSAP approach is presented based on the principles of REC which addresses the shortcomings of the original REC strategy by introducing a concept known as the Resolutions of Impact (ROI). The introduction of ROI is an attempt to address the limitations of the sliding window technique; The sliding window is explained in details in section 4.3.1. The introduction of a sliding window enables ML algorithms to be applied to time series data [76]. The ROI approach captures the length of the time period over which the time series signal reacts (in the future) to various local patterns (at present). In our evaluation, a comparative analysis is conducted which compares the forecasting capabilities of our approach with the current state of the art methods for the Irish Pig Price dataset.

4.2 Basis for Our Approach

In this section, we discuss why serial correlation, uncertainty, and long term memory provide a challenge for MSAP researchers and then articulate how these three challenges, denoted by the black circles in Fig. 4.1 influence our new MSAP strategy.

Serial Correlation. A large class of time series data are attributed by serial correlation where predicting over any steps requires the value of previous steps. The

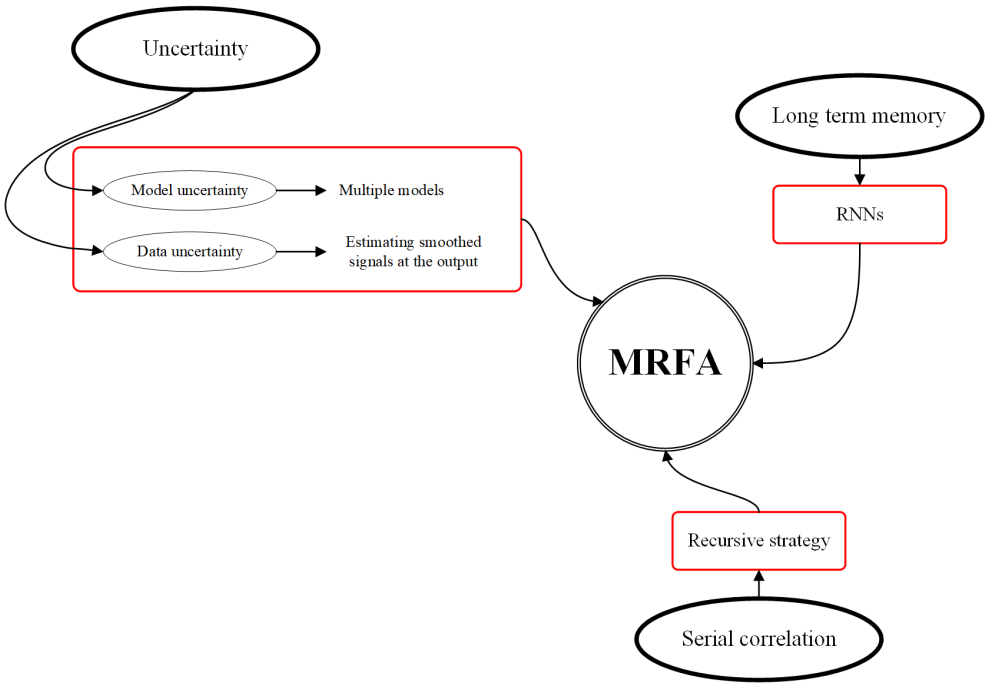


Figure 4.1: Our approach for addressing three main MSAP challenges

most primitive tool for characterizing serial correlation is the Autocorrelation Function (ACF), which is frequently used as a tool to configure ARIMA models reviewed in Chapter 2. A time series is said to have serial correlation (or more formally an autocorrelation structure) when ACF shows a strong dependence at the most recent lags specially lag one. Researchers have predominantly used autocorrelation as the basis for their predictive approach [52,66]. This suggests that serial correlation plays a pivotal role in time series prediction and *must* be considered.

Typically, in the recursive strategy or REC, predictions are estimated using multiple one step ahead predictions. When the prediction strategy moves forward in time, predicted values can be used as inputs in the prediction process. In practice, REC is implemented by training a regression model, however, a major drawback with REC is the error accumulation that accrues over longer prediction horizons. REC or the recursive strategy can be implemented using any regression model and is not limited to AR or ARIMA. As mentioned in Chapter 2, REC incorporates a number of OSAP steps and the error accumulates over time as more (previously) predicted values are used as the input to the OSAP model. In DIR, all the steps in the prediction horizon

are predicted using *the same inputs*, usually based on *actual data*. The strength of DIR is that all the steps are predicted using **actual data** and the model is not fed by inaccurate inputs (unlike in REC). However, the drawback of DIR is it discards and ignores serial correlation between lags as the prediction horizon moves forward.

Reviewing the processes of DIR and REC strategies reveals a limitation in DIR where it ignores the role of serial correlation in the accuracy of MSAP [245]. However, despite the error accumulation problem, REC has a solid strength in theory, [39]. Additionally, serial correlation between lags is one of the main pillars in the prediction process in univariate time series. As a result, we adopted the recursive strategy (REC) shown in Fig. 4.1 as the basis of our prediction strategy to account for serial correlation.

Long Term Memory. Past research has shown many processes are characterized by long term memory or values that have long term persistence (long term persistent autocorrelation structure) [79]. Long term memory differs from serial correlation as serial correlation relies on most recent time points. This means that methods that use a narrow fixed length sliding window discard or ignore the presence of long term memory in the signal and thus, make predictions based on partial information. Subsequently, methods like ANNs and SVR cannot be used for predicting long term memory processes [202]. Sliding window is a necessary component in the implementation of the recursive strategy. We will explain Sliding window in section 4.3.1.

Theoretically, RNNs (Recurrent neural networks) have the ability to predict long term memory processes as they retain a higher level of historical information. In practice, RNNs also have the potential to address the vanishing/exploding gradient problem [118, 173]. There are more advanced methods like LSTM that can be used when predicting long term memory processes but due to the large number of learning parameters in these methods, there is a significant chance of over or under fitting. This issue is further compounded if the number of available training samples is low. Based on our assessment of previous research, we adopted RNNs (shown in Fig. 4.1)

as the tool for implementing our MSAP strategy to address the long term memory issue.

Note that serial correlation and long term memory are related since long-term memory can be encoded in serial correlation. However, serial correlation mainly address the most recent lags while a long term memory may refer to a long sequence of past lags. **Uncertainty.** The presence of noise or uncertainty may impact the performance of MSAP strategies. When implementing MSAP, there are two sources of uncertainty: uncertainty that comes from the data and uncertainty introduced by the model. To mitigate the impact of the data uncertainty, researchers typically smooth the data. Smoothing will reduce the impact of outliers but can dampen patterns in the signal and reduce the the predictive power of serial correlation [94]. In our approach, we use RNN models and train each model using both original and smoothed outputs to deal with the uncertainty caused by noise (data uncertainty). However, we train our RNN models by *actual* inputs so as not to dampen serial correlation. In our approach, we assign weights to the models based on their accuracy. Therefore, if the serial correlation is very strong in a given time series, the corresponding model shows an improved accuracy and receives a greater weight, when calculating the final output.

Uncertainty of the model output incorporates uncertainties of any sources in the model, i.e, from model structure, the training algorithm, or parameter uncertainty. In order to deal with uncertainty introduced by the model, multiple models are typically employed and their results are aggregated to obtain an improved model. This resembles the function of ensemble methods where multiple methods are used to estimate the same output [91]. In our approach, we use multiple RNN models to address model uncertainty. As shown in Fig. 4.1, we incorporate multiple models to address model uncertainty and incorporate forward smoothing to address data uncertainty.

There are also two other types of uncertainty: Parameter uncertainty and forecasts uncertainty. Parameter uncertainty is a very interesting topic and can generally

be considered as a sub-problem in model uncertainty and sometimes in data uncertainty. Based on [145], parameter uncertainty can account for sampling errors, variability, and measurement errors. This definition introduces parameter uncertainty as a sub-problem in data uncertainty which is out of the scope of this research. Based on [175], Model Parameter Uncertainty accounts for the incomplete knowledge of the model parameters or inputs. When input uncertainty is the case, an input selection mechanism should be implemented prior to the training phase, which is a very interesting but challenging task. Unfortunately, in machine learning problems, input selection usually requires a large dataset and can be very time consuming, and thus is out of the scope this research due to our limitations in time and computing power. Uncertainties in model hyper-parameters can be addressed in a hyper-parameter optimization task usually implemented using a grid search mechanism which is an NP-hard problem. Forecast uncertainty is predominantly addressed using prediction intervals. Although forecast uncertainty is an important problem, it actually appears in the results. Unfortunately, there is no standard method for measuring forecast uncertainty in machine learning problems and thus, we do not study forecast uncertainty in this thesis.

4.3 MRFA Methodology

In previous section, we outlined the three challenges that the research into a new MSAP strategy must overcome, and for each challenge, we outlined our solution. However, addressing all three challenges in a single model requires focusing on the interactions between the components and an adaptation mechanism that could drive the modeling process to solve MSAP problems. In this section, we present the new MSAP strategy, known as Multi-Resolution Forecast Aggregation (MRFA), that employs the principles of REC and incorporates a new concept known as *Resolution of Impact* to address all three challenges. We first introduce Resolution of Impact, then proceed to a description of the MRFA architecture, before the model building process is described.

4.3.1 Resolution of Impact

Resolution of Impact (ROI) is the analytical engine of MRFA and plays a pivotal role in the overall approach. Using ROI for guidance, we analyze the size of the future time horizon over which local patterns have impact. The simplest representation of ROI is the sliding window (SW) technique, which is typically applied to time series to convert data into a format for suitable for machine learning prediction models. A machine learning technique requires *inputs* and *outputs* for their training procedure, and SW returns these two components for machine learning models. SW is a fixed size window moving over the time series, that returns the last unit of the covered area as the *output* and the remaining units as the *input*. ROI differs from SW in that both SW and ROI return the same data as the *input*, but the *output* returned by ROI is the smoothed data at a pre-specified resolution. This difference between SW and ROI is illustrated in Fig. 4.2.

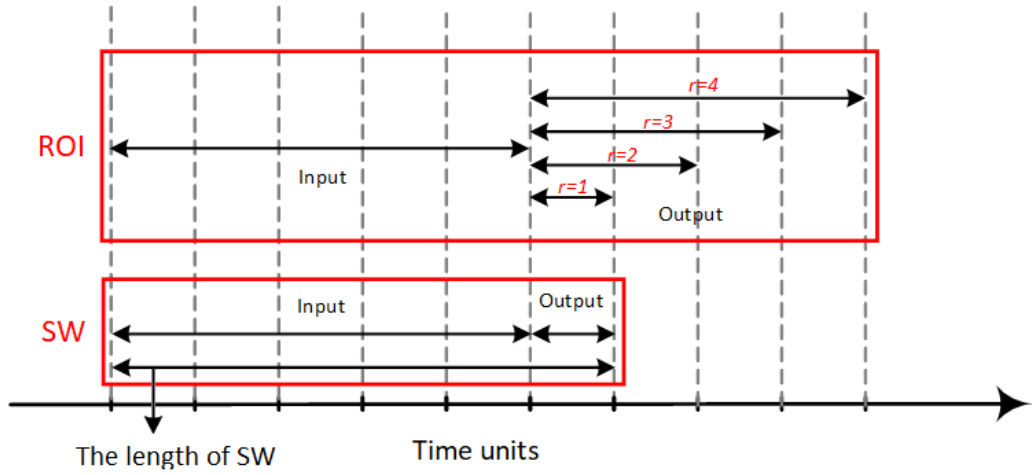


Figure 4.2: Resolution of Impact

Using Fig. 4.2, it can be seen that SW is equivalent to ROI at resolution $r = 1$. At greater resolutions, ie. $r > 1$, a smoothed signal or the average of the signal over the next r units is returned by ROI as the *output*. ROI, when analyzed at multiple resolutions, provides a simple quantitative tool for assessing memory in the signal. The simplest way to quantify memory in the signal, is to assess how many time series lags are required to be incorporated in the model in order to accurately

predict the signal. Therefore, long term memory can appear in a smoothed signal with longer resolutions . In ROI modeling, if there is a long term memory in the signal, the ROI models with longer resolutions will show higher levels of accuracy. We adopted multi-resolution analysis of ROI in MRFA to deliver an improvement in our MSAP strategy.

Note that the size of SW is fixed and chosen as a hyper-parameter, but the size of ROI can vary as determined by the resolution set being defined as a hyper-parameter. Also, in ROI, the output values are averaged to produce a new output. At first glance, it might raise an issue of discarding the autocorrelations between the outputs. However, autocorrelations are only important when used as inputs or when each output is predicted separately using a multiple-inputs multiple outputs model discussed earlier in section 2.2. Also, smoothing/averaging the outputs enables the model to deal with outliers and also to detect patterns in a larger scale.

4.3.2 MRFA Architecture and Components

By introducing the multi-resolution analysis of ROI, we provided an ability to assess the future horizon that the current pattern (the *input sequence* returned by ROI) can predict directly. To implement this, we used multiple RNN models, each addressing ROI at a different resolution. We refer to the *output* component returned by ROI as the prediction Horizon (PH). Fig. 4.3 depicts the Multi-resolution analysis of ROI using multiple RNN models.

Aside from the exploitation of memory properties in the signal, we use the multi-resolution ROI analysis shown in Fig. 4.3 to address both data and model uncertainties. Using multiple models, we can simulate the traditional approach used in ensemble methods for dealing with model uncertainty. As shown in Fig. 4.3, the time unit under PH_1 has been covered by R separate RNN models and we use the RNNs in an ensemble manner to address model uncertainty. Also, since the outputs of the RNN models are smoothed values, our approach also addresses data uncertainty. Note that the inputs to the RNN models are chosen from actual data,

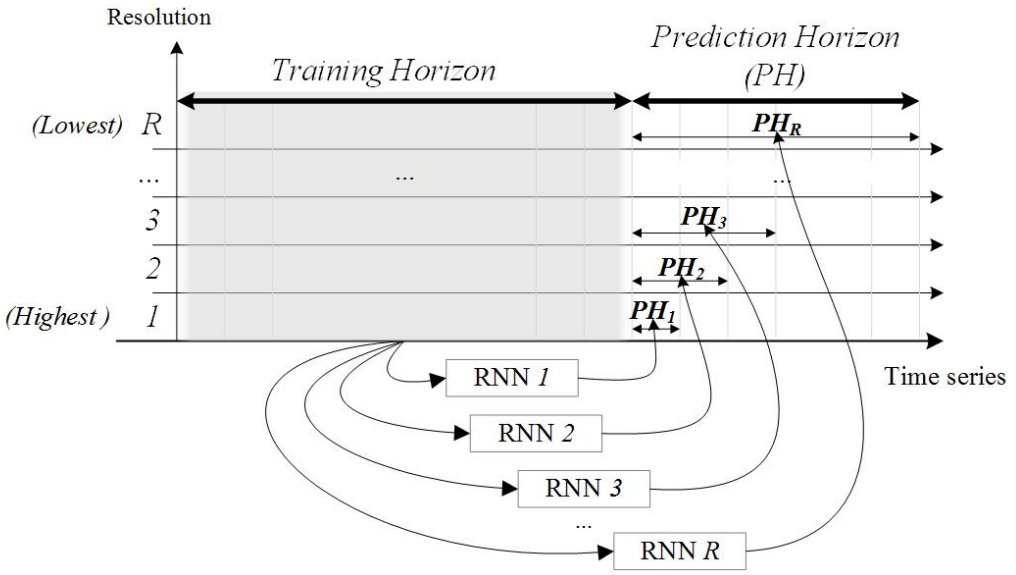


Figure 4.3: Multi-resolution ROI

which enables us to train the RNN models and build our MSAP strategy based on accurate inputs.

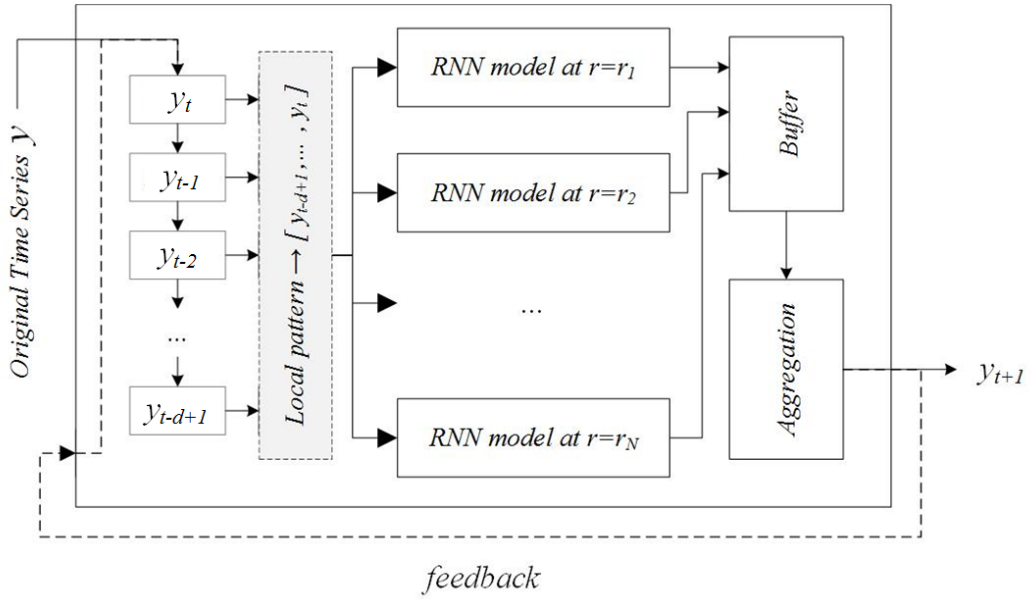


Figure 4.4: The architecture of MRFA

The schematic view of the proposed MRFA approach is illustrated in Fig. 4.4. Here, MRFA uses d time series lags, i.e, $y_t, y_{t-1}, \ldots, y_{t-d+1}$ as the *input sequence* of ROI (or the local pattern) and then uses N RNN models to analyze ROI at

N resolutions. Choosing d is an input selection problem and depending on the problem can be determined using a trial and error process. The outputs of the RNN models are then buffered for usage in further prediction steps. The *buffer* is a memory that keeps previous results and *aggregation* is an equation through which the final forecast is obtained which is explained in Eq. 4.4. The final forecasts are then calculated by aggregating weighted averages of the RNNs buffered outputs. The process of MRFA contains three main phases: training the forecast models at multiple resolutions, determining the model weights, and forecast aggregation.

Training the Forecast Models. In MRFA, for each resolution $r \in R$, where R is the set of working resolutions, a separate RNN prediction model is deployed. Resolution r defines the forecast target of an RNN model, as the mean value of the signal over the next interval of length r. In MRFA, the resolution determines the forecast target of the corresponding RNN model while, for every RNN, the input is fed from the time series of highest resolution, i.e., the signal itself. An RNN model of resolution r is trained to perform a one step prediction in resolution r from the past values of the time series of highest resolution. For multi-step ahead forecasting, a feedback loop is conducted in the MRFA model, as illustrated in Fig. 4.4. As shown in Fig. 4.4, we reduced the effects of error accumulation encountered in the recursive strategy by incorporating a multi-resolution analysis of ROIs in MRFA.

As shown in Fig. 4.4, the feedback loop feeds the RNN model by a delayed version of the forecasts, for further estimates until the entire PH is forecasted. For the time series y_t, the output of the RNN model which is modeling the time series at resolution r is denoted by (PH_r) in Eq. 4.1.

$$\underset{\mathbf{r}\in\mathbf{R}}{PH_r}(t) = \frac{1}{r}\sum_{i=1}^{r} y_{t+i} \tag{4.1}$$

Determining Weights.

The MRFA model assigns weights to the outputs of the RNN models. The weight reflects the influence of the specific model in the corresponding resolution. Reliability

can be determined by the model performance which represents the model's accuracy for a single step ahead prediction problem. In [23], a reciprocal value of nMSE is used as model weight. For a single step forecasting model, the weight at resolution r i.e. W_r is calculated using Eq. 4.2, where $nMSE_r$ represents the accuracy of the RNN model which is employed at resolution r.

$$W_r = \frac{1}{nMSE_r} \tag{4.2}$$

$$nMSE = \frac{1}{n_{Test}} \sum_{i=1}^{n_{Test}} (\frac{y_i - \hat{y}_i}{Y_{Max} - Y_{Min}})^2 \tag{4.3}$$

The nMSE parameter for the RNN model for modeling signal Y is calculated using Eq. 4.3, where Y_{Max} and Y_{Min} are the maximum and minimum value of Y, and y_i and $\hat{y}_i$ are the *actual* and *estimated* value of Y respectively, for the i^{th} test sample. n_{Test} is the number of out of sample forecasts and thus, in Eq. 4.3 nMSE is calculated for n_{Test} test samples $i = 1, .., n_{Test}$. Weights in Eq. 4.2 do not need to sum to 1, because (as shown later in Eq. 4.4) our aggregation approach standardizes the final forecast by incorporating the weights (W_r) in its denominator.

Forecast Aggregation.

In this phase, the forecasts made by the RNN models at different resolutions are aggregated to provide forecasts for the entire PH. For every TU in the PH, a set of candidate forecasts are introduced by the RNN models. At a specific resolution, the final forecast is obtained by aggregating the candidate values introduced by the RNN models at different resolutions through weighted averaging in Eq. 4.4, where $\hat{y}_r(t+1)$ is the forecast made by the RNN model of resolution r.

$$\hat{y}(t+1) = \frac{\sum_{r \in R} W_r \sum_{i=0}^{r-1} \hat{y}_r(t+1-i)}{\sum_{r \in R} rW_r} \tag{4.4}$$

Eq. 4.4 demonstrates how multiple time points contribute to the prediction estimate. This is because in MRFA, a RNN where $r > 1$ provides a partial forecast for more than a single step. Therefore, as long as r covers the delay i in $\hat{y}_r(t + 1 - i)$, the corresponding forecast can be used to improve the accuracy. Also, as shown in Eq. 4.4, we use previous predictions in order to assist in current predictions. We do this because more recent predictions are less accurate with the recursive strategy.

4.4 MRFA Evaluation

We now turn our attention to the validation of the MRFA model which is presented in 2 parts now. Firstly, we present a sensitivity analysis which his necessary to fine tune hyper parameters and then proceed to discuss our comparative evaluation.

4.4.1 MRFA Parameter Settings

In this section, we tackle the issue of hyper parameter settings for our evaluation. Our goal is to examine all possible combinations of hyper parameter selection in order to obtain the optimal configuration. These settings are crucial to our comparative evaluation in the following section. This step includes a sensitivity analysis on the significant levels of the parameters reported in MRFA analysis.

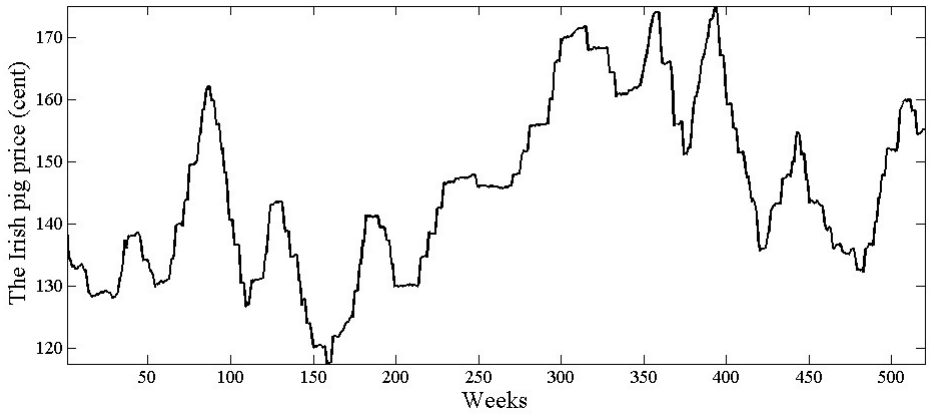


Figure 4.5: The Irish pig price

The performance of MRFA is highly dependent on the performance of its RNN models and its resolutions. In this study, the same RNN configuration was employed in each experimental cycle. MRFA was studied for 7 resolutions ($N = 7$ in Fig. 4.4) ranging from $r_1 = 1$ to $r_7 = 7$ in size, meaning 7 RNNs were implemented in every experiment. Two parameters were identified as having major contributory factors to the performance of the RNN models: (1) the number of lags being fed as inputs into RNNs, i.e. delays, and (2) the RNN's degree of recurrence that determines the highest number of steps that the neurons in the RNN's hidden layer can remember its own previous outputs. The recurrence delay parameter determines the number of previous outputs for each neuron in the hidden layer to be included in the calculation of the final output. Fig. 4.6 presents a sensitivity analysis and shows the performance of MRFA for different values of recurrence delay versus delay (time series lags).

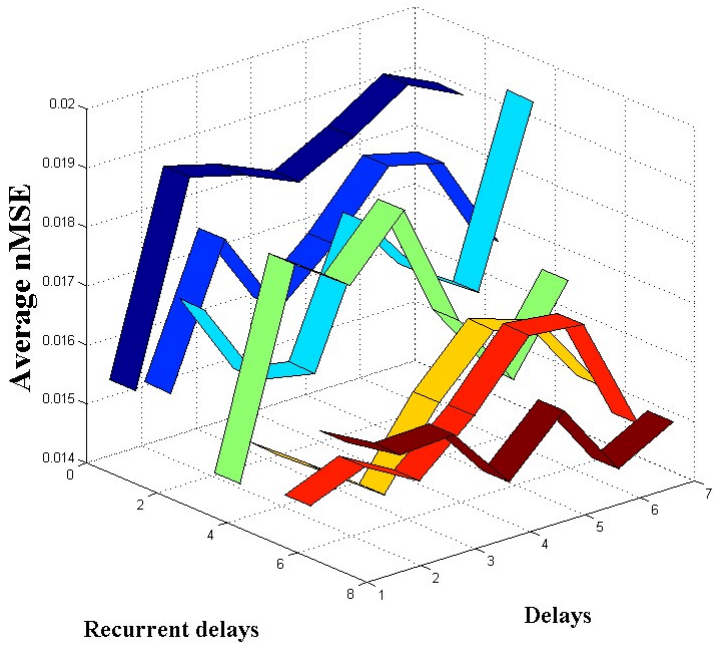


Figure 4.6: Sensitivity analysis: Recurrence delay versus time series lags

As PH (the lengths of the prediction horizon) grows longer, more uncertainties are added to MSAP. Fig. 4.7 illustrates the MRFA's accuracy with respect to increase in PH. The results in Fig. 4.7 shows that an increase in PH (the length of the

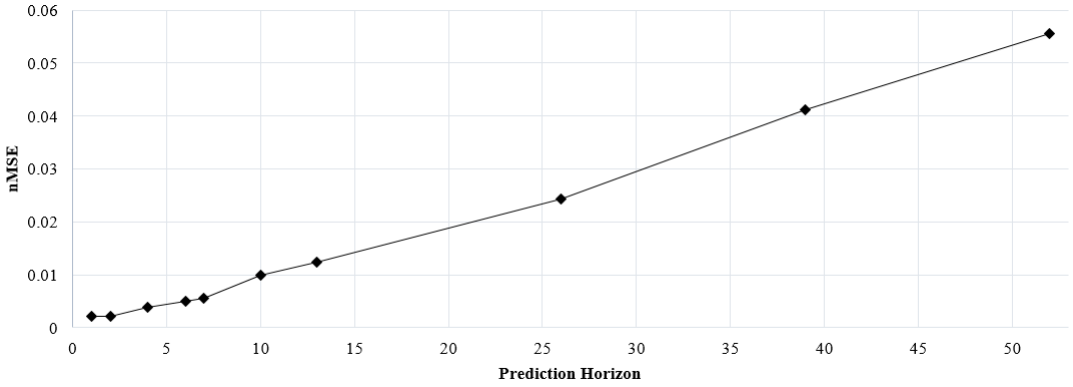


Figure 4.7: Accuracy of MRFA with respect to prediction horizon

prediction horizon) lowers the accuracy of predictions.

4.4.2 Comparative Analysis

For the MRFA evaluation, six state of the art methods have been chosen for comparative analysis: ARIMA [46], NN [117], RNN [151], DIR [17], MIMO [43], and ARIMA-NN [183]; DIR and MIMI were both implemented using ANN. The prior sensitivity analysis on the NN model suggests a combination of 12 hidden neurons and a sliding window of size 10. The sensitivity analysis indicated that the RNN model exhibits minimum error variance when characterized by 10 input delays, 6 recurrence delays, and 12 hidden neurons. The ARIMA parameters of the ARIMA-NN models are also identified by analyzing Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots, introducing ARIMA(p=1,d=1,q=1) as the appropriate model; where p is the order of the AR component, q is the order of MA model, and d is the differencing order. The performance of the NN model for modeling residuals was affected by the size of the sliding window and the number of hidden neurons. Sensitivity analysis on these factors demonstrates that the minimum error variance is reached when the sliding window is of size 13 and the hidden layer contains 10 neurons. Performance comparisons between MRFA, ARIMA, NN, RNN, DIR, MIMO and ARIMA-NN with respect to growth in PH are illustrated in Fig. 4.8. In these experiments, 5 percent of the data were used as the test set

in a bootstrapping procedure and the rest were used as training. Note that due to the need for a period of 52 samples for each MSAP test experiment and also a continuous set of samples for the training set, no more than 5 percent could be separated for the test set. Otherwise, the training performance could be reduced due to the lack of adequate training samples.

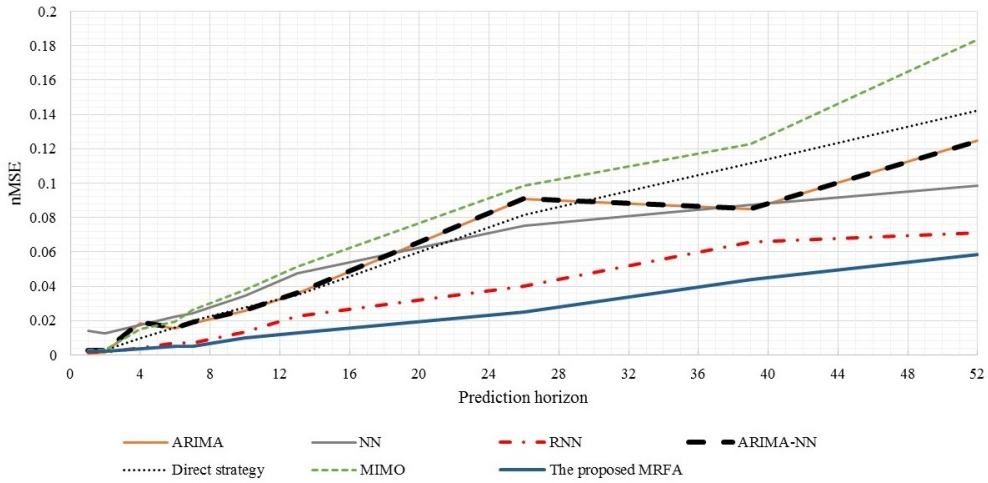


Figure 4.8: PH comparison between MRFA and ARIMA, NN, RNN, DIR, MIMO, and ARIMA-NN

Fig. 4.8 shows how MRFA outperforms ARIMA, ARIMA-NN, RNN, and NN at every PH. The comparison also reveals that, as PH increases, there is an increase in prediction error. However, this increase occurs more slowly for MRFA than ARIMA, ARIMA-NN, RNN, DIR, MIMO, and NN.

20 different time series were chosen for analysis in this research and were taken from disparate monthly, weekly, daily and hourly data sources. The monthly series were: lake Erie levels (1921-1970), monthly milk production in pounds or `m_m_p` (1962-1974); and the number of persons employed in Australia (1978-1991). The weekly time series were: two Irish beef prices (2011-2018); Irish pig prices, 2007-2016; German pig prices (2008-2016); Canadian barley (2008-2016); and German feed barley (2008-2016). The daily time series were: foreign exchange rates (1979-1998); minimum temperatures in Melbourne (1981-1990); total female births in California (1959-1959); mean temperature of the Fisher River (1988-1991); bike share variables

(2011-2012); and births in USA between 1978 and 2015. Hourly data was taken from one series measuring hourly carbon dioxide emissions. In these experiments, for each time series, 10% of the data has been chosen to make out of sample forecast as the test set and the experiments were conducted in a bootstrapping process; The rest of the data was used for training purposes.

In order to characterize each dataset, differing tests and metrics were applied. The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots [47] were analyzed to confirm the existence of seasonality. Conditional Heteroscedasticity was examined using the Ljung-Box Q test [47] on the squared residual series. Also, the Hurst exponent and Detrended Fluctuation Analysis (DFA) were conducted on the data to characterize memory in the time series. The Hurst exponent and DFA are explained in details in Chapter 5, section 5.3.1. Table 4.1 shows these characteristics as well as the frequency, sample size and type for each of the original time series. Type is determined based on DFA analysis, explained in detail in section 5.3.1. Gaussianity is evaluated using D'Agostino's K^2 test [69] and denoted by H for high, L for low, V for very low, and 0 for no Gaussianity.

Table 4.1 presents the results for 8-step-ahead prediction using MRFA and the recursive strategy implemented by NN, SVR, ARIMA, RNN and LSTM. In this table, m_l_e_l, m_m_p, and m_n_o_e indicate lake Erie water levels, monthly milk production pounds, and the level of employment in Australia, respectively (Monthly data). H_Um_3P and S_Um_2P indicate two Irish beef prices, IreCent is the Irish pig price, GerCent is the German pig prices, CanBar is Canadian barley prices and GerFBar is the German feed barley prices (weekly data). Daily foreign exchange rates are denoted by d_f_ex_r, daily minimum temperatures in Melbourne by d_m_t, daily total female births in California by d_t_f_b, mean daily temperature of Fisher River near Dallas by m_d_t_f, daily bike share variables by d_b_sh1, d_b_sh2 and d_b_sh4, a US economic series by Ec_unem, births in US in 1978 and 2015 by US_B_78 and US_B_15, (Daily data) and hourly carbon dioxide emission by LNOxEm.

Results and Discussion.

Table 4.1: Experimental results

ID	Frequency	Time series	Gaussianity	Sample size	Seasonality	Stationarity	CH	Hurst exponent	DFA	Sample entropy	Type	Min error
A01	M	m_l_e_l	L	600	12	N	Y	0.59	1.17	0.88	Non-stationary	MRFA
A02	M	m_m_p	L	156	12	N	N	0.69	1.52	0.68	Brownian noise	MRFA
A03	M	m_n_o_e	V	759	12	N	Y	0.97	1.28	0.29	Non-stationary	NN
A04	W	H_Um_3P	H	357	52	N	Y	0.93	1.79	0.41	Non-stationary	MRFA
A05	W	S_Um_2P	L	357	52	N	Y	0.92	1.68	0.60	Non-stationary	SVR
A06	W	IreCent	V	512	52	N	Y	0.92	1.86	0.32	Non-stationary	MRFA
A07	W	GerCent	V	468	52	N	Y	0.86	1.58	0.53	Non-stationary	NN
A08	W	CanBar	V	468	52	N	N	0.88	1.64	0.20	Non-stationary	LSTM
A09	W	GerFBar	V	468	52	N	N	0.86	1.63	0.25	Non-stationary	NN
A10	D	d_f_ex_r	V	4774	365	N	Y	0.95	1.52	0.05	Brownian motion	SVR
A11	D	d_m_t	V	3650	365	Y	N	0.90	1.08	1.62	Pink noise	ARIMA
A12	D	d_t_f_b	V	365	-	Y	Y	0.61	0.69	2.20	Stationary	NN
A13	D	m_d_t_f	V	1461	365	N	N	0.88	1.24	0.73	Non-stationary	MRFA
A14	D	d_b_sh4	V	731	-	N	Y	0.51	0.66	2.05	White noise	ARIMA
A15	D	d_b_sh2	V	731	-	N	N	0.72	1.05	0.88	Pink noise	MRFA
A16	D	d_b_sh1	V	731	-	N	N	0.77	1.07	0.87	Pink noise	MRFA
A17	D	Ec_unem	V	574	-	N	N	1.00	1.58	0.23	Brownian noise	MRFA
A18	D	US_B_15	V	365	-	N	N	0.09	0.08	0.65	RNN	RNN
A19	D	US_B_78	V	365	-	N	N	0.23	0.29	1.21	Anti-correlated	LSTM
A20	H	LNOxEm	0	8081	24	Y	N	0.39	0.37	0.45	Anti-correlated	ARIMA

For each time series shown in table 4.1, the method with the minimum prediction error (column **Min error**) is reported as *the preferred method*. The prediction error is calculated as the average nMSE. For each series, the order of the $SARIMA(p, d, q)$ model was obtained using the maximum likelihood using a Kalman filter [116]. A detailed discussion on optimal models is beyond the scope of this research and is expanded upon in detail in [279, 291]. The experimental results based on the application of the methods discussed are presented in Table 4.1.

We can see there are relationships between the preferred method (column **Min error**) and time series characteristics in Table 4.1. Obviously, drawing strong conclusions requires studying a relatively larger number of time series. However, based on our experiments with the 20 series in Table 4.1, the following insights can be

drawn:

- Sample entropy of values close to zero indicate low levels of complexity, and thus, higher predictive probabilities. For A19, A14, A12, and A11, the sample entropies were relatively high with the best performance exhibited by ARIMA (2 series), NN and LSTM. However, we expected ARIMA to outperform all the other methods on low sample entropy series, since such series are often predictable and ARIMA performs well on highly predictable data.

- MRFA is suitable method for predicting Brownian motion (Brownian motion will be explained in detail in section 5.3.1), as MRFA was the preferred method on 2 of the three Brownian motion series (A02, A10 and A17). Brownian motion is a type of self-similar processes which is identified using DFA analysis which is explained in details in section 5.3.1. We expected LSTM to show a good performance on Brownian noise as LSTM has the ability to predict complex series. However, MRFA is based on RNN models which are structurally similar to LSTM and we believe that for this reason, MRFA showed comparable performance on these time series.

- MRFA is a good method for predicting Pink noise (Pink noise will be explained in detail in section 5.3.1), as MRFA outperformed on 2 of 3 pink noise time series (A11, A15 and A16). Pink noise is an unusual characteristic in time series as it presents a trade-off between the probable predictability of Brownian motion and the complete unpredictability of white noise. Our interpretation is that MRFA delivered the best performance due to the incorporation of several RNNs, as they are able to capture complex dynamics in nonlinear time series.

- ARIMA and NN are approaches suited to predicting stationary time series (indicated by stationary and white noise in column **type**). It suggests that ARIMA and NN which are characterized by relatively smaller sample entropy (compared to RNN, LSTM and MRFA) are predominantly suitable for series which have a steady mean, variance and auto-correlation. However, white noise (series A14) is an unpredictable process and this explains why this result

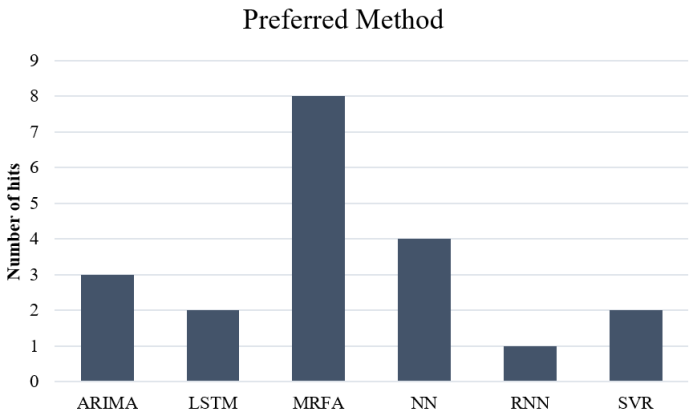


Figure 4.9: Methods and Performance

as reliable as could be expected.

- These results demonstrate that MRFA is a good choice of model when Gaussianity is significant, as in 6 out of 16 series which the distribution of their fluctuations is highly similar to Gaussian distribution (H and V), MRFA was the preferred method.

- NN is an appropriate method when dealing with CH, as NN outperformed the other methods for these series. We expected LSTM to show higher performance as LSTM was specially designed to process complex events, such as CH, in time series data.

Fig. 4.9 outlines the number of times each method was chosen as the preferred method. There is strong evidence that the performance of different methods are a function of the time series characteristics involved. The evidence also suggests that MRFA is a robust performer as it is either the *preferred* method or was one of the *high performers* on the majority of the series examined with a preferred candidate score of 50%. In particular, it was the preferred method on 80% of the series when applied to series with either Brownian or pink noise and scored 44% when applied to data with non-stationary. MRFA, NN and LSTM all appeared to perform well when non-stationarity or CH existed within the data. As would be expected, ARIMA performed well with non hetroscedastic data, and is a much

easier method to implement in comparison to machine learning approaches such as MRFA, NN, RNN or LSTM.

In summary, we demonstrated where MRFA showed superior performance over other methods which involved many aspects. However, the major drawback with our analysis is the small size of the testing benchmark; it only contained 20 time series. This leads to a final conclusion that a more robust validation will require a far higher number of time series and from this, we should be able to extract a deeper understanding of the behaviour of both our model and those that are used in a comparative analysis.

4.5 Summary

In this Chapter, we presented our recursive approach which we have named, Multiple Resolution Forecast Aggregation, an approach that addressed the shortcomings of the original REC model by using the *Resolutions of Impact* concept. We examined the efficacy of this approach using 20 time series in a study on the behaviour of Irish Pig Price data. Our evaluation showed that the preferred method for certain models depends on the *characteristics* of the underlying time series. As such, a large number of time series which exhibit a more diverse range of characteristics is required to generate stronger predictive models. However, our literature review suggests that a large dataset with this guarantee of *diversity* does not exist. This provides a clear requirement for the next step in our research: an approach to construct a large number of time series with the appropriately diverse range of characteristics.

Chapter 5

Establishing Diversity in Synthetic Time Series

In the previous Chapter, we motivated the requirement for a methodology to generate a large number of time series which comprised a rich level of *diversity* in terms of time series characteristics. In this Chapter, we introduce a framework to both generate the time series data and to validate its diversity. In section 5.1, we provide a background to this aspect of our research before discussing time series characteristics in section 5.2 and presenting the methodology for constructing time series in section 5.3. We then present evaluation metrics in section 5.4 before discussing the results of our evaluation in section 5.5. Finally, in section 5.6, we present our conclusions to this Chapter.

5.1 Background and Motivation

When assessing the performance of a Time Series prediction method or strategy, researchers are typically faced with the challenge of identifying appropriate datasets for training and testing purposes. Typically, the researcher will use a dataset that has originated from their specific domain of interest, or if the cost of data collection

is prohibitive, they will attempt to generate synthetic/simulated data that reflects the characteristics of the identified problem. In order to avoid costly data collection, synthetic data has been used in many differing fields of research in place of real or live data [159]. It has also been shown to be useful when attempting to evaluate proposed methods [133], assist in data imputation [259] or supplement training datasets when machine learning approaches require a sufficient volume of data [208]. Public repositories of real data have been established by researchers in an attempt to provide domain-specific examples (or several domains) among which Kaggle and UCI are the most well-known [1, 2].

Data generation processes mainly focus on simulating training data for specific problems where data samples share a set of domain-specific characteristics. These characteristics are modified with a random component whose distribution is characterised within the domain and thus, synthetic time series will typically share the same time series characteristics [182]. However, in order to evaluate the performance of a time series method, *diversity* has been shown to be the main requirement in training datasets [177]. Unfortunately, in the available time series repositories, diversity has not been promoted as an underlying characteristic. As shown in Chapter 2, previous research on time series generation has focused on enhancing the availability of data, with simulated data sets consisting of data series with similar characteristics. In this research, the focus on the application of new and existing time series prediction techniques requires that any proposed approach be tested on a variety of diverse series. The overall intention is to understand the range of characteristics that suit any particular method. Generating diverse datasets and subsequently implementing the proposed and existing time series prediction algorithms on these synthetic time series will allow researchers to assess the breath of application for their proposed approach.

To find the strengths and the weaknesses of an algorithm, its performance should be evaluated against diverse types of data [22, 60]. Researchers have used several time series data repositories such as Kaggle, UCI, NN5 and M5 [124]. However, these repositories do not provide a wide and diverse range of time series and thus,

prevent researchers from examining the appropriateness of their proposed algorithm under a variety of conditions. Many time series in the M5 collection contain missing values for which features such as entropy cannot be measured. In addition, applying detrending and deseasonalizing on the M5 series cannot be done with 100 % accuracy since their trend and seasonality models are not available; this leaves unknown fluctuations on the remainder which leads to inaccurate evaluation of features. At present, stress testing time series algorithms with datasets that have a diverse range of time series characteristics, has not received significant attention in the literature. Also, even forecasting competitions don't pay attention to the diversity of time series characteristics. The aim of this step in our research is to develop an approach that will create a large dataset that comprises a wide range of disparate time series. These time series can then be used by the scientific community to establish the appropriate application area of any future prediction algorithms.

Contribution Performance evaluation is an essential component in the development of time series prediction algorithms. The choice of datasets used to test a new approach can have a considerable impact on its perceived applicability in a real world environment. Typically, when there is a lack of appropriate datasets, researchers have generated artificial datasets during their testing phase. In time series analysis this strategy has been applied but has predominantly focused on generating similar time series with moderate fluctuations in the datasets. In this research, we present a time series generation algorithm that creates sets of time series with a diverse range of characteristics. In particular, we focus on time series with long term memory and stationarizable irregularities.

In this Chapter, we present a framework that generates a set of disparate time series for **time series prediction** purposes. By diversity, we mean different combinations of time series features and thus, a diverse time series collection is expected to contain time series representing different possible combinations of features. Time series data usually consist of four components: Trend, Seasonality, Cyclicality and Irregularity. The majority of studies conducted on time series prediction emphasize the necessity of stationarity and thus, valid time series prediction has generally

been implemented with detrending and de-seasonalizing operations as mandatory pre-processing steps [290]. Also, Cyclicality is typically unpredictable and often removed when performing de-seasonalizing or detrending. Thus, the remainder (i.e., irregularity) can be stationarized in order to apply predictive analyses. The majority of past studies assume that the irregular component is a white noise and contain random fluctuations, such as [109, 187, 188, 221, 265]. However, many studies have shown that irregularity can carry important information, and retain long term memory [253]. Therefore, based on the definition of irregularity [253], we conclude that the irregular component is expected to be stationarizable, carry information, have long term memory and exhibit random fluctuations. With our approach, we simulate irregularity using fractional Browning motion (fBm), which is a stationarizable random process and characterized by long term memory. We also introduce a new measure for assessing diversity and as a result, improve the quality of assessment.

5.2 Time Series Characteristics

Fundamentally, a time series is composed of *trend* (T), *seasonality* (S), *cyclicality* (C) and *irregularity* (I) components [51]. Trend carries the information associated with long term or low frequency behavior of the series. Many time series exhibit a regularly repeating pattern known as *seasonality*, often under the influence of external periodic drivers such as seasons, weather or holidays. Both seasonality and *cyclicality* are the result of repeated patterns, however, the difference is that seasonality has *constant* repeating intervals whereas cyclicality repeats over *non-equal* intervals [146]. There is also the concept of multi-seasonality which will be incorporated later in Eq. 5.5. The *Irregular* component is the residual signal, when the trend, seasonality and the cyclical components are removed. Typically, the irregular component is assumed as a patternless signal that only presents random fluctuations [253].

The functional form of each time series is based on how these components are generated and combined. The degree or level of the presence of each item is known as the

profile of the time series [51], where each time series component can be described by its own individual time series.

Trend is often described as the change in the mean of the signal over time [48]. Practitioners in disparate fields have differing opinions on how trend should be interpreted. The most common interpretation is that trend is a reflection of the overall scale and magnitude of the signal. Many studies, identify the low frequency component of the signal as the trend by decomposing the frequency domain signal [41]. However, other studies have obtained trend by eliminating local fluctuations using smoothing [48].

Seasonality is typically identified by regularly repeating patterns. Seasonality often occurs under the influence of external periodic drivers such as seasons, weather or holidays. One needs only generate a regularly repeating pattern over equal intervals to simulate seasonality [48].

Cyclicality is a long-term, upward or downward curve which usually represents irregular swings. In most business and economic time series, cyclicality appears in periods of many years (maybe decades) and thus, is often regarded as part of long term trend. In financial time series, cyclicality occurs as broad swings around the trend line [93], and differs from seasonal variations in that the length of time period under consideration is not constant. In other words, seasonal variations can mainly be anticipated, while cyclical variations are often considered to be unpredictable [112].

The **Irregular** component I is the part of the time series that cannot be described by trend, cyclicality or seasonality and is measured as the residual of the signal after removing trend, seasonality and cyclicality. Although irregularity is traditionally considered as random and pattern-less fluctuations, more recent studies believe that it contains important information which does not appear in long-term or repetitive structures [253].

5.3 Methodology

In this section, we introduce our approach for generating synthetic time series. In order to generate synthetic time series, one needs to first *create* each time series component and then *combine* them. Thus, a time series generation process has two phases: *component generation* and *the combination of components*. Since the characterization of the non-equal intervals of cyclicality is not always possible, the research community suggests that cyclicality and trend can be addressed as a joint element i.e., the *trend-cycle* component [40, 115, 290]. We will first discuss how we generate different types of each component and in the following section, the methods by which these components are combined are described. In the rest of this section, we will first present our approaches for simulating the Trend-cycle component, seasonality and irregularity in §5.3.1. Then, in §5.3.2, we will present our models for combining these components.

5.3.1 Simulating Time Series Components

We now describe the functions used to simulate each of the three time series components. We begin by presenting different *Trend-Cycle* functions; then show how *Seasonality* is introduced; and finally, we present functions for *Irregularity*.

Trend-Cycle Components In general, trend can be linear or non-linear. A **linear trend** is simulated using Eq. 5.1 where a is a constant showing the slope of the linear line. A linear trend is shown in Fig. 5.1.

$$y = a \times t + b \tag{5.1}$$

where b is the y intercept.

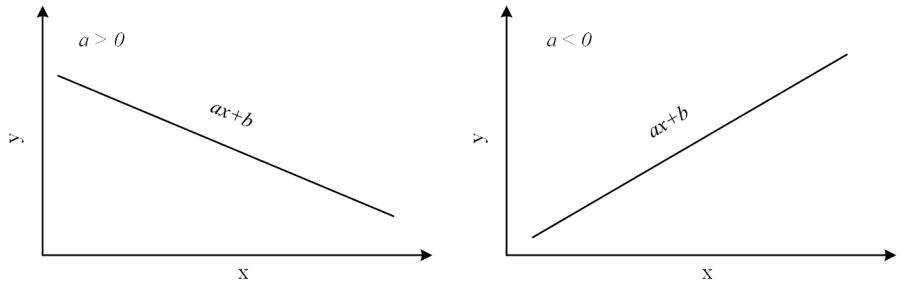


Figure 5.1: A linear trend

The shape and the boundaries of a **non-linear trend** depend on the problem domain. In this research, two general functional forms have been considered to generate non-linear trends. The first is a piece-wise linear function representing a trend changing direction at k pre-specified points, yet remaining linear between each two consecutive points as shown in eq. 5.2. The second form is the trend-cycle component presented in [123]. Eq. 5.2 has a curve consisting of n changing points $p_1 < p_2 < \ldots < p_n$, between each consequent pair showing a linear behavior but with a different slope $a_1, a_2, \ldots, a_n$, where $b_1, b_2, \ldots, b_n$ are constant.

$$y = \begin{cases} a_1 t + b_1 & ; \quad 0 \leq t < p_1 \\ a_2 t + b_2 & ; \quad p_1 \leq t < p_2 \\ & \ldots \\ a_n t + b_n & ; \quad p_{n-1} \leq t < p_n \end{cases} \tag{5.2}$$

In our implementation, $p_1, p_2, ..., p_n$ are fixed values chosen between 1 and the length of the time series. An assumption made in this research is that the trend is a continuous curve so that at $t = p_i; i < n$, no abrupt fall or rise occurs in the curve, shown as a constraint in Eq. 5.3.

$$a_i(t = p_i) + b_i = a_{i+1}(t = p_i) + b_{i+1} \tag{5.3}$$

Fig. 5.2 depicts the constraint of continuity represented in Eq. 5.3.

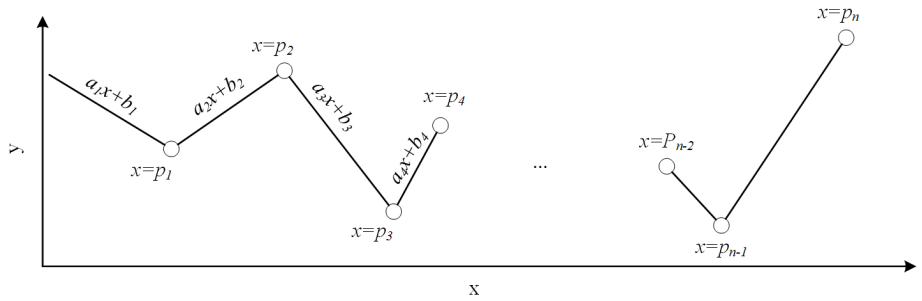


Figure 5.2: The piece-wise linear trend

The *cyclical* component can be added into the trend-cycle component through the introduction of a second order long term seasonal component [51]. Here, the trend-cycle component is created using Eq. 5.4 which uses a *sinusoidal* function both with and without multiples of a linear function, with α is a constant.

$$\begin{cases} y = a \times sin(\alpha t) & \text{simple} \\ y = at \times sin(\alpha t) & \text{multiplied by a linear function.} \end{cases} \tag{5.4}$$

Note that the *sin* function should be implemented as a low frequency signal, with a large value of α, in order to simulate long term effects in the trend-cycle component.

A simple sinusoidal trend is shown in Fig. 5.3.

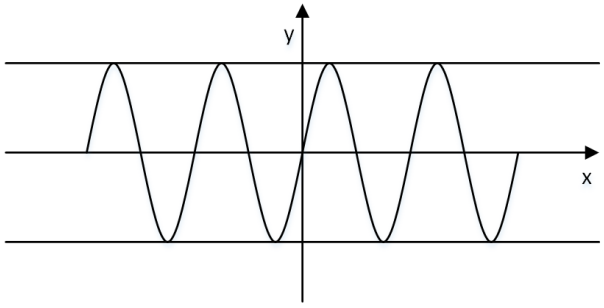


Figure 5.3: A simple sinusoidal trend

A sinus function multiplied by a linear trend is shown in Fig. 5.4.

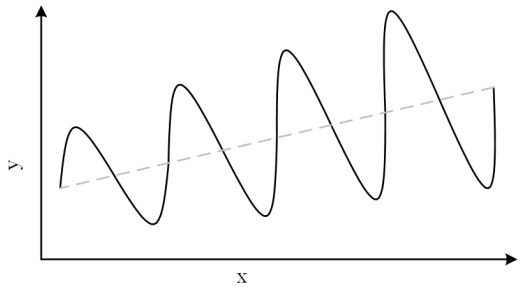


Figure 5.4: A sinus function multiplied by a linear trend

Seasonality Components. In our approach, it is assumed that seasonality can be simulated by impulse, step-wise, triangular or a combination of sinusoidal functions of differing scales and phases; where smoothing is conducted randomly to avoid passing sudden big changes. Sinusoidal patterns can be simulated using one or a combination of multiple sinusoidal functions as shown in Eq. 5.5, where, β_1 and β_2 are the weights; α_1 and α_2 are the phases for the sinusoidal functions; and y_t the time series. Note that $\alpha_0, \alpha_1, \beta_0$ and β_1 are constants.

$$y_t = \beta_0 sin(\alpha_0 t) + \beta_1 sin(\alpha_2 t) \tag{5.5}$$

A **Step-wise** pattern is a type of latch function (with two stable states) that changes between two values at fixed intervals shown in Eq. 5.6, where p ($p \geq 0$) is the period, t is time and m is an integer.

$$y_t = \begin{cases} 1 & 2mp < t < 2mp + 1 \\ 0 & 2mp + 1 < t < 2mp + 2 \end{cases} \tag{5.6}$$

The step-wise function is shown in Fig. 5.5.

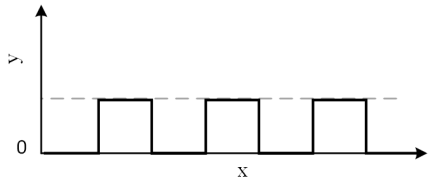


Figure 5.5: The Step-wise function

The *Triangular* function is similar to the step-wise function where the step-wise effect occurs in the slope of the line. Eq. 5.7 implements the triangular function with a fixed slope α, where b_0 and b_1 are constants, p is the period, t is time, m is an integer.

$$y_t = \begin{cases} \alpha t + b_0 & 2mp < t < 2mp + 1 \\ -\alpha t + b_1 & 2mp + 1 < t < 2mp + 2 \end{cases} \tag{5.7}$$

The Triangular function is shown in Fig. 5.6.

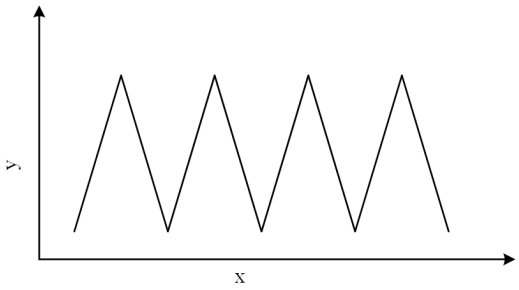


Figure 5.6: The Triangular function

Impulsive seasonality has a discrete pattern that has a value of 1 at fixed intervals and 0 otherwise. The implementation is shown in Eq. 5.8, where t is time, and p is the period.

$$y = \begin{cases} 1 & \lfloor t/p \rfloor = t/p \\ 0 & otherwise \end{cases} \tag{5.8}$$

The Impulsive function is shown in Fig. 5.7.

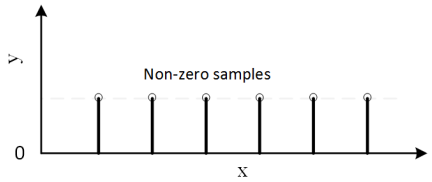


Figure 5.7: The Impulsive function

Note that the Impulsive function only has one non-zero value over each interval, while the step-wise function switches on-off (1/0) at the start of each interval. **Irregularity Components.** The irregular component can be studied in terms of complexity and memory (or correlation). Providing a formal definition for time series complexity is a difficult and challenging task. Past studies have tried to explain complexity as the degree of disorder (randomness) or unpredictability in the signal [200]. Memory reflects how consistently the signal relies on its past: if not affected by or contaminated with random fluctuations. If this component is not formed by a completely random process or under the influence of unknown external random forces, the assumption is the presence of Long Range Dependence (LRD) which is caused by a memory buffer [33].

$$\rho(k) \approx c_\rho |k|^\delta \tag{5.9}$$

Eq. 5.9 is used to determine if a process has LRD, where k is the number of lags, c_ρ is a positive constant and $0 < \delta < 1$. Thus, a process has LRD when the sum of autocorrelations $\rho(k)$ decays to 0 slowly. When there is a memory in the signal we can expect that with increase in the number of lags (k), the sum of the autocorrelations for the k lags do not fade to zero immediately. Otherwise, no information from the past values are carried over the autocorrelations.

The Hurst exponent H [250] is one of the most popular methods to measure LRD. H attempts to explain LRD as a property of stochastic self-similar processes. $x(t)$ is self-similar with the Hurst exponent H, when for a stretching factor λ the rescaled

process $x(\lambda t)$ is equal to the original process $x(t)$ in terms of distribution as in Eq. 5.10, where $\doteq$ denotes *equality* in terms of distributions.

$$x(t) \doteq \lambda^{-H} x(\lambda t) \tag{5.10}$$

If the fluctuations are stationary (the process has a constant mean and a constant variance), the process is said to have fractional Brownian motion (fBm). Based on [33], the auto-correlation function for fBm processes is defined in Eq. 5.11.

$$\rho(k) = \frac{1}{2}(|k+1|^{2H} - 2\,|k|^{2H} + |k-1|^{2H}) \tag{5.11}$$

Based on [250], applying a first-order Taylor expansion to $\rho(k)$ from Eq. 5.11 delivers the functionality in Eq. 5.12, for $k \to \infty$.

$$\frac{\rho(k)}{H(2H-1)|k|^{2H-2}} \to 1 \tag{5.12}$$

It can be inferred from Eq. 5.12 that the autocorrelation $\rho(k) \propto |k|^{2-2H}$ when $H > \frac{1}{2}$, based on [250].

Eq. 5.12 explains the behaviour of the irregular component using the well known Taylor method [33]. Since the Hurst exponent (H) is the core of the equation and is an input parameter to our time series generation function 5.12, we will provide a brief overview of its usage.

The Hurst exponent [121] was proposed to model the cyclic behavior of Nile floods and is conceptually close to Brownian motion, where temporal fluctuations are independent and the standard deviation σ at step n scales proportionally as $\sigma \propto n^{\frac{1}{2}}$. The study found that temporal fluctuations are not independent and show a pattern of a power law with an exponent over $\frac{1}{2}$, a non-stationary process equivalent to fBm. The fBm processes are characterized by long range dependence, i.e, future fluctuations are influenced by past fluctuations and the sign of movements are less likely to

change frequently. However, the phenomenon is fundamentally has a probabilistic formation where oscillations preserve the stochastic structure of the quantity, yet exhibit long-term memory. In such processes, the scaling of the standard deviation is proportional to $\sigma \propto nH$.

Note that such processes possess a probabilistic-statistical nature and thus, from some point on, the signal will saturate but slightly fluctuate around a quite low value. However, long term memory remains a general property of the process. Eq. 5.9 is not only bounded to the non-stationary process of fBm, but can also explain the stationary process of the fractional Gaussian noise (fGn) shown in Eq. 5.13, where B_H is an fBm with a Hurst exponent equal to H.

$$fGn_H(t) = B_H(t+1) - B_H(t), \tag{5.13}$$

In order to accommodate various theories, where the irregular component is treated as fractional Brownian motion (either with a zero-mean or carrying information), three major noise model types were incorporated:

- fractional Gaussian noise (fGn), which represents stationary series with a constant mean and variance;
- fractional Brownian motions (fBm), which are non-stationary series with time-dependent variance [160];
- multi-fractal Brownian motion (mBm), which is considered for the case where the Hürst index H is a function of time t.

fGn, fBm and mbm are frequently used by researchers to simulate real world systems due to their similarity to natural processes. The Fractional Brownian motion family appear as a very natural object as it has the three of the following characteristics: [172]: continuous Gaussian, self-similarity and stationary fluctuations. To simulate fGn, fBm and mBm, this study uses the Davies-Harte algorithm [74]. In Craigmile [68], the authors demonstrated that the Davies-Harte can be used to generate such

stationarizable Gaussian processes. We used the Davies-Harte method to generate fGn, fBm and mBm in our research.

5.3.2 Combining Time Series Components

It is important to note that there is no standard way of combining these components to generate a time series. Studies such as [81] consider the irregular component as the base and manipulate it by adding trend and seasonality to create time series data. In our time series construction method, we consider all possible additive and multiplicative combinations of *trend-cycle* T_t^c, *seasonality* S_t and *irregularity* I_t, using the approach presented in [123]. There are 8 possible models for combining T_t^c, S_t and I_t, shown in Table 5.1.

Table 5.1: Time Series Component Combinations

	Model	Description
1	$Y_t = T_t^c + S_t + I_t$	The additive model
2	$Y_t = (T_t^c + S_t)I_t$	Trend-Seasonality additive multiplied by the Irregularity
3	$Y_t = (T_t^c + I_t)S_t$	Trend-Irregularity additive multiplied by the Seasonality
4	$Y_t = (S_t + I_t)T_t^c$	Seasonality-Irregularity additive multiplied by the Trend
5	$Y_t = T_t^c S_t + I_t$	Trend-Seasonality multiplicative added to the Irregularity
6	$Y_t = T_t^c I_t + S_t$	Trend-Irregularity multiplicative added to the Seasonality
7	$Y_t = S_t I_t + T_t^c$	Seasonality-Irregularity multiplicative added to the Trend
8	$Y_t = T_t^c S_t I_t$	The Multiplicative model

In Table 5.1, Model 1 is a pure additive model, and is the most widely used model in the time series community. Model 8, is a multiplicative model, and is the second most popular model among time series researchers. The other models in Table 5.1 are also used in the time series studies with *Model 3* and *Model 5* being more popular because they incorporate irregularity I_t using an addition operation. In this research, all 8 combinations are implemented during time series construction.

5.4 Validation Metrics

The literature has outlined a variety of metrics that allow the researcher to understand the characteristics of the series. The following have been used extensively in the literature:

- Long-Range Dependence
- Complexity
- Fisher Information
- Normality

Long-Range Dependence measures the degree of dependence (correlation) over long intervals of time, and indicates the level of "memory" in a time series. LRD can be assessed using the Hurst exponent and Detrended Fluctuation Analysis (DFA). DFA is a more systematic method for assessing LRD in comparison to the Hurst exponent. Based on the evaluation of DFA on LRD, correlation can be categorized into six well known classes, which are shown in Table 5.2.

Table 5.2: Interpretation of DFA

Value	Type
$0 < \alpha < 1/2$	Negatively-correlated
$\alpha \simeq 1/2$	Uncorrelated, white noise
$1/2 < \alpha < 1$	Correlated
$\alpha \simeq 1$	1/f-noise, pink noise
$\alpha > 1$	Non-stationary, unbounded
$\alpha \simeq 3/2$	Brownian noise

The output of the DFA analysis or α can be interpreted as follows (based on Table 5.2): $\alpha = 1$ indicates perfect (self) similarity in the data [113] ; $\alpha = 1/2$ represents no similarity (or no memory); $1/2 \leq \alpha \leq 1$ describes positive correlation, with similarity (memory) increasing with the values of α; $\alpha \leq 1/2$ indicates negative

correlation; $\alpha > 1$ indicates that while correlations exist, they cannot be described in the form of a power-law relationship. A special case where $\alpha = 1.5$, indicates Brownian noise or the integration of white noise. α also provides information about the roughness of the time series where larger values of α belong to smoother time series. $1/f$ noise can be interpreted as a compromise between the complete unpredictability of white noise (very rough landscape) and the very smooth landscape of Brownian noise.

An important advantage of DFA is that it allows for the detection of long-range correlation in non-stationary time series. Alternatively, the Hurst Exponent has been the traditional metric for LRD [121], and divides time series data into three categories: Negatively-correlated $0 < \alpha < 0.5$, Uncorrelated $\alpha \simeq 0.5$ and Correlated $0.5 < \alpha < 1$.

The **Complexity** of a time series can be evaluated using Entropy measures [226]. In this research, a number of entropy measures have been implemented, including Shannon entropy, Spectral entropy and SVD entropy.

- **Shannon entropy:** For a signal y with sample size N, sample entropy is calculated by Eq. 5.14, which is the negative logarithm of the conditional probability that a sub-series of length m matches point-wise with the next point with tolerance (distance less than) r [223].

$$H(Y) = -\sum_{i=1}^{N} p(y_i) \log p(y_i) \tag{5.14}$$

- **Spectral entropy:** Spectral entropy is calculated based on Shannon entropy, and quantifies the spectral complexity or the randomness of the power spectrum of the time series over a long period of time [216]. Spectral entropy is calculated by Eq. 5.15.

$$Hs = -\sum_{k=1}^{N} P_k \ln(P_k) \tag{5.15}$$

where k represents the frequency λ_k and P_k is calculated by Eq. 5.16

$$P_k = \frac{|\lambda_k|^2}{\sum_i |\lambda_i|^2} \tag{5.16}$$

In Eq. 5.16, $|\lambda_k|^2$ is the Fourier power spectrum of the signal at frequency λ_k.

- **SVD entropy:** SVD entropy is an indicator of the dimensionality of the time series, i.e. the number of eigenvectors needed for an adequate explanation of the given time series [9]. the SVD-entropy of the time series y is calculated by Eq. 5.17.

$$E(X) = -\frac{1}{\log(N)} \sum_{j=1}^{N} V_j \log(V_j) \tag{5.17}$$

In Eq. 5.17, V_j is a normalized eigenvalue of the matrix XX^T

$$V_j = -\frac{s_j^2}{\sum_k s_k^2} \tag{5.18}$$

where s_j^2 is an eigenvalue of the matrix XX^T.

X is a matrix that is obtained by conducting a delay-embedding operation known as the Hankelization of the time series $H_\tau(x)$, shown in Eq. 5.19 with the delay parameter τ, [95].

$$H_\tau(x) = \begin{bmatrix} y_1 & y_2 & \cdot & y_{N-\tau+1} \\ y_2 & y_3 & \cdot & y_{N-\tau+2} \\ \vdots & \vdots & \ddots & \vdots \\ y_\tau & y_{\tau+1} & \cdot & y_N \end{bmatrix} \tag{5.19}$$

Fisher information (FI) [225] is a measure of information content in data. FI quantifies the amount of information the data represents about an unknown parameter and determines how much information can be obtained about an unknown parameter from a specific amount of data.

For the time series X and its associated density function $f_{y,\theta(y)}$ that depends on the parameter $\theta \in \Theta$ and θ_k the k-th element of θ, FI is calculated in our implementation using Eq. 5.20

$$FI(f_{y,\theta})_{\theta_k} = \int f_{y,\theta}(y) \left(\frac{\partial \ln f_{y,\theta}(y)}{\partial \theta_k} \right)^2 dy \tag{5.20}$$

FI is a function of the variability of the data in such a way that variability has an inverse relationship with FI, meaning that high variability leads to low FI [224]. Also, in contrast to Shannon entropy that is a general measure of smoothness, FI provides a local measure of smoothness, since FI is calculated based on the derivative of the probability distribution function (PDF) [54]. Therefore, in comparison with Shannon entropy, FI has more sensitivity to perturbations that affect the PDF. A considerably disordered time series exhibits a uniform (or unbiased) PDF that is wide and smooth and thus, is an indication of unpredictability [54]. This is an indication of low predictability or low values of FI. In contrast, a time series with a solid structure (or a small degree of disorder) shows bias to certain states and the PDF is steeply sloped about these states, leading to a high FI.

Normality tests are used to determine whether a data set follows the normal distribution. In this research, we use Kurtosis, Skewness and the Gaussianity of the Differences (GoD) to measure the normality of the generated time series [90]. *Kurtosis* measures the number of outliers in the dataset with respect to a normal distribution: when Kurtosis is high, the dataset has a higher number of outliers (heavy tail in the distribution); when kurtosis is low, the outliers are low to none (light tail). *Skewness* measures the symmetry of the distribution: when positive, the distribution has a longer or fatter tail on the right side; when negative, the left side of the distribution has a longer or fatter tail; when zero, the distribution is symmetrical. *GoD* measures the normality of the distribution of the first lag difference (change) of the time series. In this research normality was measured using the Shapiro-wilk test [220].

5.5 Evaluation

5.1.

In this section, we examine the disparity of 53,637 time series each has a length between 400 and 45,000 samples that were generated using the approach outlined in section 5.3. To create each time series, first, the implementation method of each time series component (Trend-cycle component, seasonality and Irregularity) is chosen. For example, Trend-cycle component= Linear-sinusoidal, seasonality = step-wise and irregularity= fractional Brownian noise. In the next step, coefficients and constants in the implementations of these components are specified. At the end, the created time series components are combined into a single time series using a model chosen from Table 5.1. Initially, we use individual histograms to graphically represent the diversity of the generated time series for each feature. We then evaluate the diversity of the combined set of features using the multivariate entropy score presented in [214]. Finally, we propose an alternate metric which measures the *coverage* of the dataset within the metric feature space. A number of examples of the generated time series are depicted in Fig. 5.8.

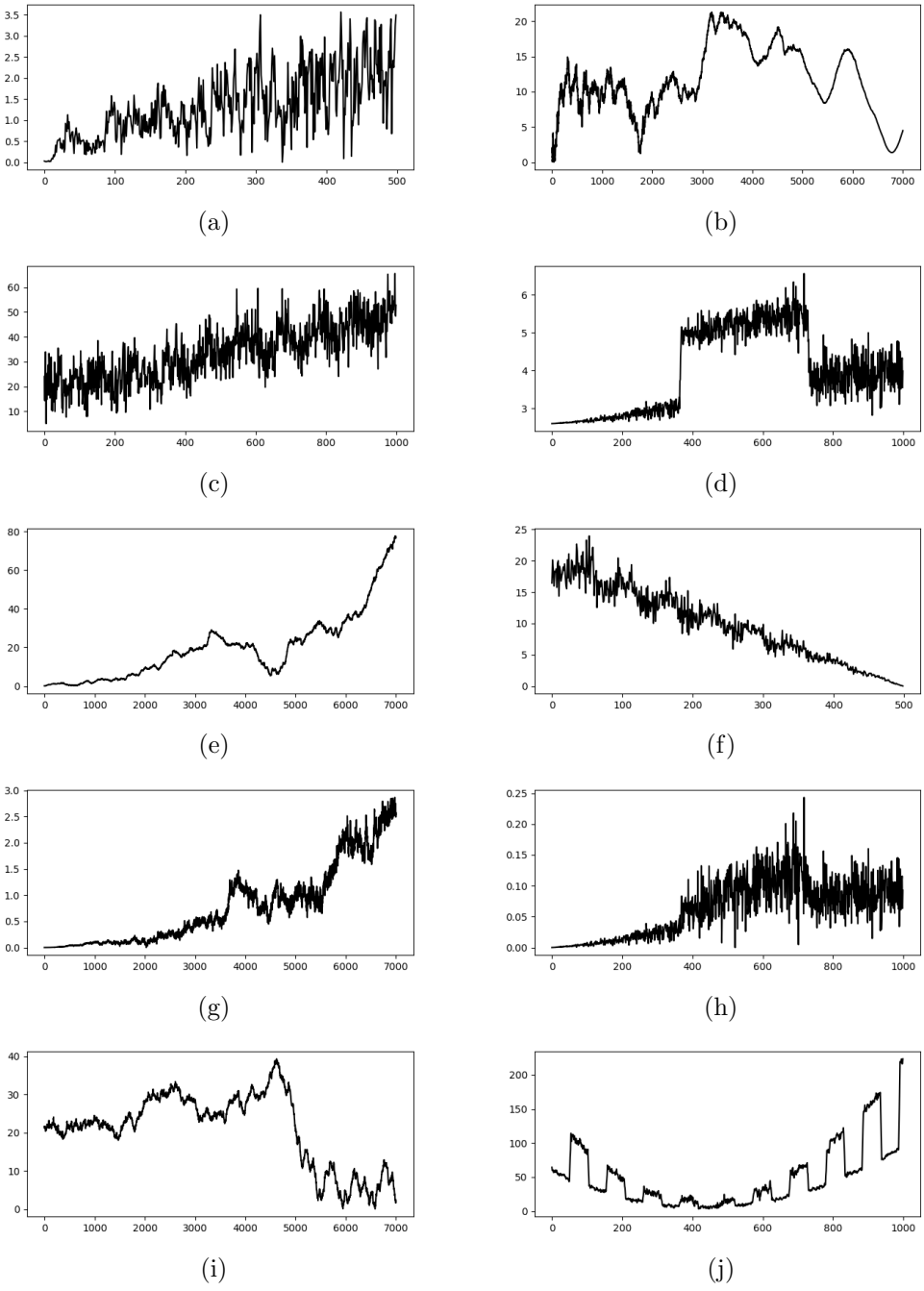


Figure 5.8: Examples of the generated series

One approach to improve this section can be to compare the diversity of our series with that of famous time series repositories such as Kaggle. However, some of our features such as Spectral entropy require stationarity as a pre-requisite prior to

the implementation of the algorithm. Also, we used DFA analysis which requires a sufficiently long time series in order to produce valid results [254]. In [8], the authors suggested that a time series of 256 samples is insufficient for reliable application of DFA analysis. Our experiments showed that many of kaggle time series cannot be stationarized by a simple differencing and need to be studied for stationarity, individually. Also, a large number of kaggle time series are short series for which DFA analysis cannot offer accurate results. An appropriate study on Kaggle time series requires conducting various stationarization strategies as well as dealing with the small sample size problem in DFA analysis. Due to some limitations such as time and computing power and man power, performing experiments on kaggle time series is beyond the scope of this research. We will consider this as a priority step in the future works.

5.5.1 Visualizing Diversity using Histograms

The diversity of the generated time series can be visualized using histogram graphs. A histogram is a two dimensional bar chart that represents the distribution of data over a continuous interval where each bar displays the frequency of the data at the corresponding interval.

The Detrended Fluctuation Analysis (DFA) method was applied to the generated series in order to calculate LRD with the LRD histogram shown in Fig. 5.9. This demonstrates that the generated synthetic series contain negatively-correlated $0 < \alpha < \frac{1}{2}$, white-noise $\alpha \simeq \frac{1}{2}$, positive correlation $\frac{1}{2} < \alpha < 1$, Pink noise $\alpha \simeq 1$, Brownian motion $\alpha \simeq \frac{3}{2}$ and unbound non-stationary time series and thus, encapsulate all forms of long range dependence. Note that our interpretation from diversity is not to have a uniform distribution of values over $(0, 2)$, but we expect to have at least one sample for each bin in the histogram shown in Fig. 5.9.

We evaluate the complexity of the generated series using Shannon, Spectral and SVD entropies. Fig. 5.10 shows that for the generated series, Shannon entropy can take a value between 8.6 and 14.5, and 95% of the time series have an entropy

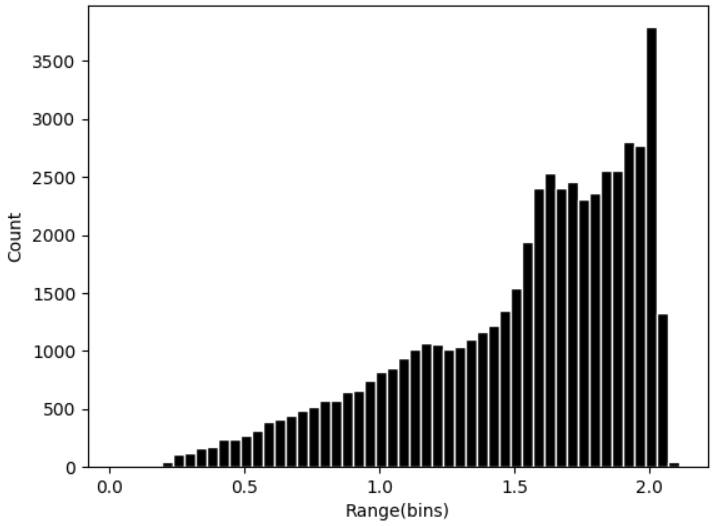


Figure 5.9: Results for LRD Metric (DFA)

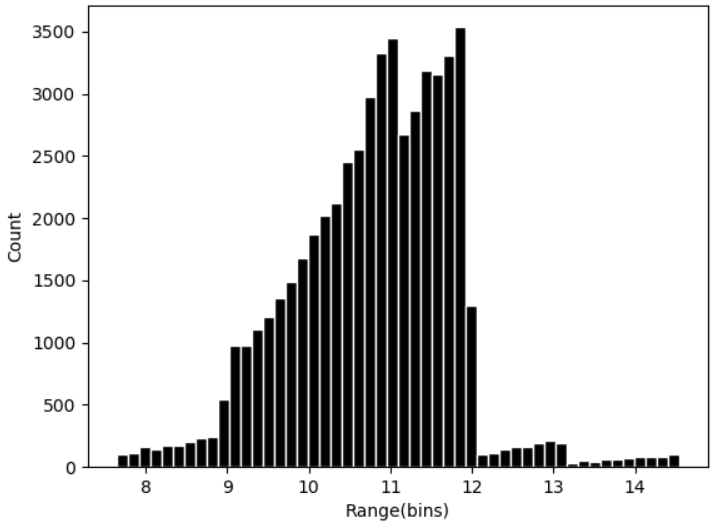


Figure 5.10: The histogram of Shannon entropy

between 9 and 12.

The histogram of the Spectral entropy is shown in Fig. 5.11. In practice, based on [255], spectral entropy *SEnt* represents the uniformity of the power spectrum distribution and greater values report that the power spectral distribution of the

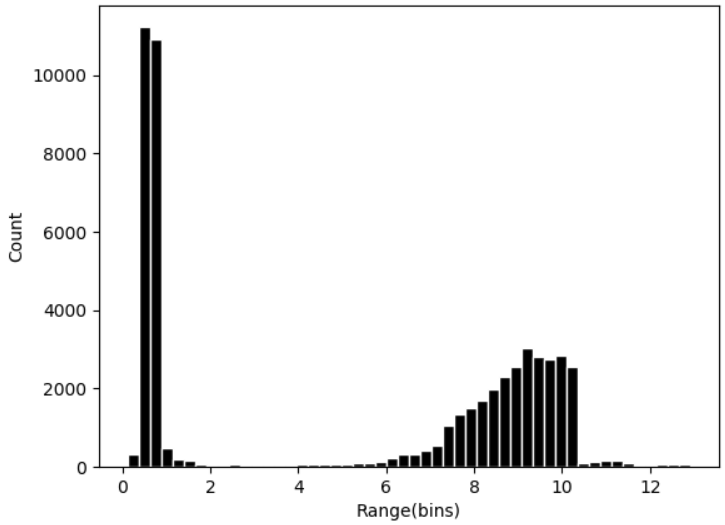


Figure 5.11: The histogram of Spectral entropy

signal is closer to the uniform distribution. Fig. 5.11 shows that a large number of the series show low spectral entropy $0 < SEnt \leq 1$ and also a large number of series show a very high spectral entropy $1 < SEnt$.

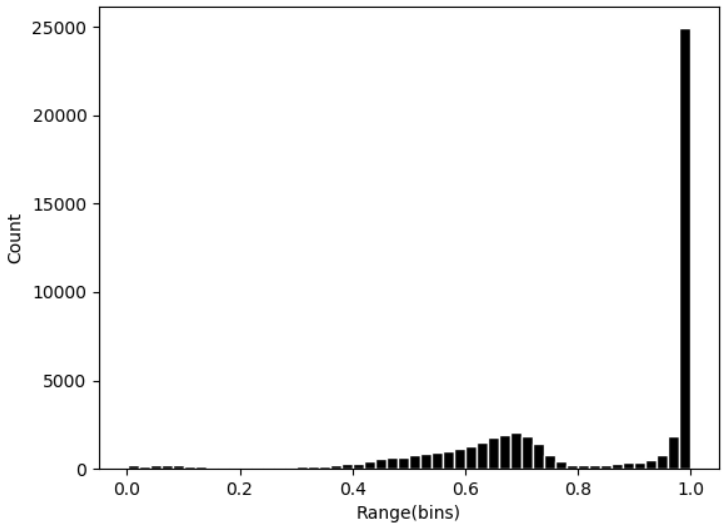


Figure 5.12: The histogram of SVD entropy

The results shown in Fig. 5.12 suggest that almost 50% of the generated series are

very complex (SVD Entropy $\approx$ 1); some series are slightly complex (low values of SVD entropy correspond to non-uniform distribution of the singular values which is an indicator of low-complexity); and the rest of the series have mid-to-high degree of complexity. Therefore, Fig. 5.12 shows that we have low-complex, mid-complex and highly complex series. Although the number of series in these three categories are not equal and the spread is not uniform, but our interpretation from diversity is to have at least one time series in each category.

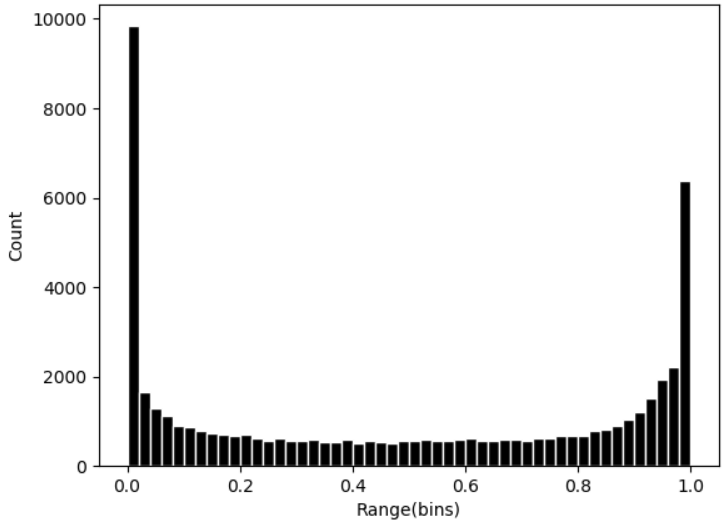


Figure 5.13: Results for Stationarity

The distribution of the p-values reported by the Dickey-Fuller stationarity test is shown in Fig. 5.13. A $p - value \leq 0.05$ indicates that the series is stationary and results show that a significant number of series $\approx 40\%$ fall into this interval. There are also numerous series with a $p - value \geq 0.05$ demonstrating that the generated series contain both stationary and non-stationary time series.

In Fig. 5.14, the Kurtosis results show the expected diversity of negative (series of light tails or series of no outliers), zero (occasional outliers) and positive values (series of heavy tails or series with significant or numerous outliers). This is a strong indicator of diversity across the datasets.

Skewness results are presented in Fig. 5.15, and show a large number of time series

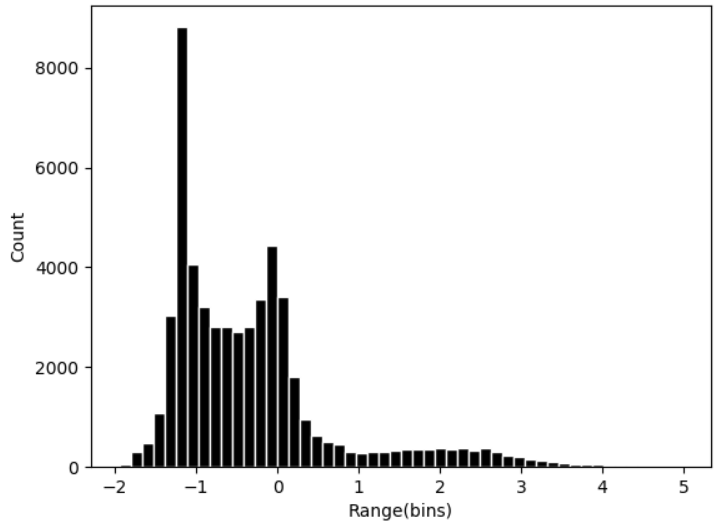


Figure 5.14: Results for Normality (Kurtosis)

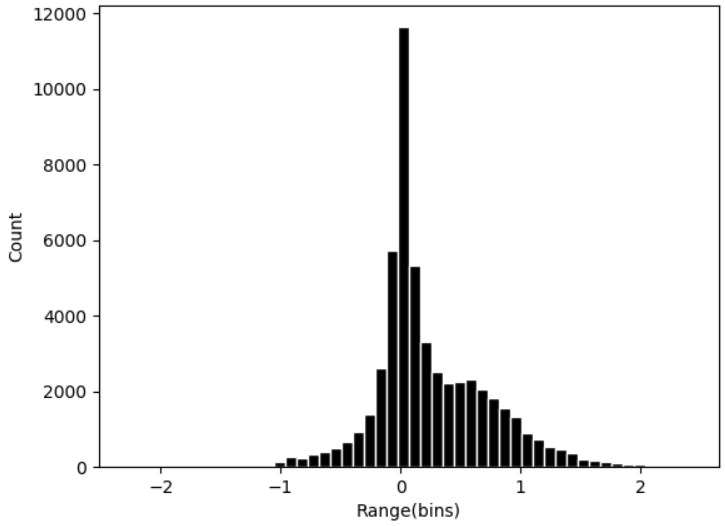


Figure 5.15: Results for Normality (Skewness)

with negative skewness (series with a fatter or longer tails on the left side), zero skewness (series of symmetrical distribution) and positive skewness (series of heavy or long tails on the right side). Once again, this indicates a high level of diversity across the datasets.

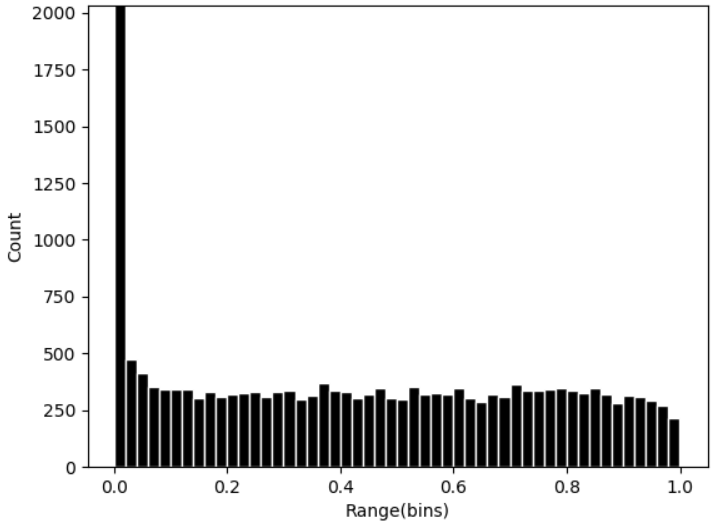


Figure 5.16: Results for Normality (GoD) or the p-values p-value from Shapiro Wilk test

Fig. 5.16 illustrates the distribution of the gaussianity of the differences. A p-value of 1 indicates that the series follow a normal/Gaussian distribution and a p-value of 0 indicates non-normality. The results show the generated series cover the entire range between zero and complete normality and thus, demonstrates a high level of diversity for the generated series.

The histogram of the Fisher information metric for the generated time series is shown in Fig. 5.17. The histogram of the Fisher information shows that for a large number of series, FI is close to zero. This indicates that these series are characterized by highly complex structures in the series and thus, exhibit a limited degree of predictability. The figure also shows that the dataset covers a wide range of values between 0 and 1 and thus, the dataset contains time series of various degrees of predictability.

Summary. Overall, the histogram graphs provide visual evidence of the diversity of individual features. Our results indicate that the synthetic time series show an acceptable diversity over individual features. Additionally, the entropy metrics indicate that our series are generally of high complexity.

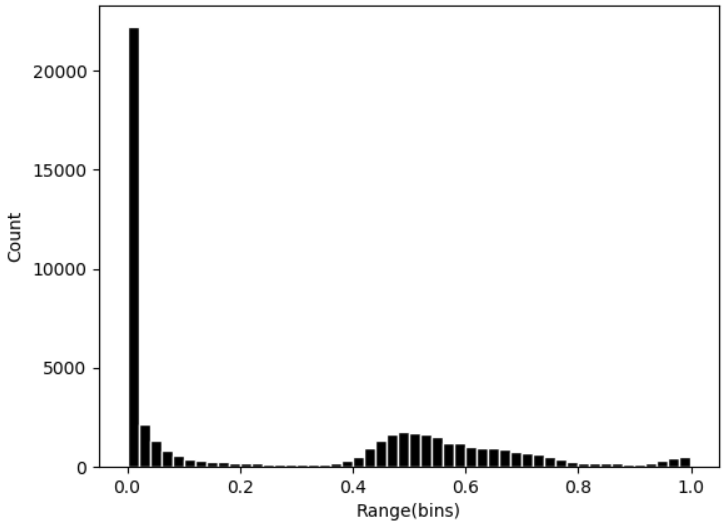


Figure 5.17: Results for the Fisher Information

5.5.2 Multivariate Entropy Score

For this part of our evaluation, we employed a Multivariate Entropy Score for diversity, that incorporates multiple evaluation metrics of the entire synthetic dataset. The Multivariate Entropy Score is based on Shannon's entropy function which is frequently incorporated to measure the amount of information in an encoded message [214].

$$H(X) = -\sum_{i=1}^{S} p(x_i) \log p(x_i) \tag{5.21}$$

The function is shown in Eq. 5.21, where $x_1, x_2, ..., x_S$ are the possible values of X, $p(x_i)$ is the probability of observing x_i or $X = x_i$ and S is the number of categories for an individual metric.

Our interpretation of diversity is interpreted as a measure of *evenness* [214], which provides a normalized value for $H(X)$ based on its maximum, and shown in Eq. 5.22.

$$H_{max}(X) = -\sum_{i=1}^{S} \frac{1}{S} \log \frac{1}{S} = \log S \tag{5.22}$$

Therefore, the *diversity* of feature X is calculated by Eq. 5.23.

$$H_E(X) = \frac{H(X)}{H_{max}(X)} = -\frac{1}{\log S} \sum_{i=1}^{S} p(x_i) \log p(x_i) \tag{5.23}$$

As shown in Eq. 5.23, $H_E(X)$ is obtained by dividing $H(X)$ by $H_{max}(X)$ which is a type of standardization of $H(X)$ to take a value between 0 and 1. As a result, the diversity of feature X is explained as a value between zero and one.

Assuming that all features have the same significance, the diversity for a multivariate (multi-feature) dataset with k features can be obtained using Eq. 5.24, where H will range between 0 (zero diversity) and 1 (maximum diversity).

$$H = \frac{1}{k} \sum_{i=1}^{k} H_E(X^k) \tag{5.24}$$

Due to the presence of a correlation between some of the metrics, only 5 metrics were chosen to be used for assessing the diversity including: Spectral Entropy, Kurtosis, Skewness, GoD and DFA. In order to implement this metric, each feature was categorized into buckets/zones that have been traditionally used by researchers. The categorization of the features, later shown in Table 5.3, are as follows:

- Spectral Entropy was categorized into three categories including A:$X < 1$, B:$1 \leq X < 9$ and C:$9 \leq X$.
- Kurtosis was categorized into three categories including A:$X < -0.3$, B:$-0.3 \leq X < 0.3$ and C:$0.3 \leq X$, based on [228].
- Skewness was categorized into three categories including A:$X < -0.3$, B:$-0.3 \leq X < 0.3$ and C:$0.3 \leq X$, based on [228].
- GoD was categorized into two categories including A:$X < 0.02$ and B:$0.02 \leq$

X.

- DFA was categorized into seven categories including A:$X < 0.45$, B:$0.45 \leq X < 0.55$, C:$0.55 \leq X < 0.95$, D:$0.95 \leq X < 1.05$, E:$1.05 \leq X < 1.45$, F:$1.45 \leq X < 1.55$, G:$1.55 \leq X$.

Table 5.3 shows the breakdown of the proportion of series that belong to each of the categories outlined above. An N/A implies that this category is not appropriate for that metric.

Table 5.3: Proportion of dataset relative to time series characteristic

Feature	A	B	C	D	E	F	G
Spectral Entropy	0.42	0.28	0.29	N/A	N/A	N/A	N/A
Kurtosis	0.59	0.25	0.15	N/A	N/A	N/A	N/A
Skewness	0.07	0.58	0.35	N/A	N/A	N/A	N/A
GoD	0.70	0.3	N/A	N/A	N/A	N/A	N/A
DFA	0.017	0.012	0.092	0.035	0.190	0.067	0.584

Table 5.4: H scores for each metric

Feature	$H(X)$	H_{max}	H_E
Spectral Entropy	1.55	1.58	0.98
Kurtosis	1.366	1.58	0.86
Skewness	1.25	1.58	0.79
GoD	0.88	1.00	0.88
DFA	1.837	2.80	0.65

Table 5.4 shows the H_{max} and H_E of each metric for the full dataset. These interim results are used to calculate the diversity as our final evaluation is to measure the diversity and coverage rate. Here, $H(X)$, H_{max} and H_E which were obtained using equations 5.21, 5.22 and 5.23, and show that the level of diversity for each of the metrics examined ranges between 0.65 for DFA to 0.98 for Spectral Entropy. Based on Table 5.4 and Eq. 5.24, the overall diversity score H for the dataset was **0.83**. The obtained diversity , ie. **0.83**, is very close to 1 (as the maximum possible value)

and far from zero (as the value for a zero diversity), which demonstrates that our dataset exhibits a significant level of diversity.

5.5.3 Metric Space Coverage

The feature space for the data is identified as all potential category combinations of the metrics outlined above in Table 5.4. Apart from the the *evenness* aspect of diversity which was measured in the previous section, diversity must be measured in terms of the *coverage* of the feature space. This issue has not received attention in the literature and in our view, must be considered when aiming for a diverse dataset. Assume that feature $f \in F$, has n categories. For a feature set F of size M (M is the number of features), where each feature f_i has n_i categories, the number of possible combinations (or the feature space size) f_S is obtained by Eq. 5.25.

$$F_S = \prod_{i=1}^{m} n_i \tag{5.25}$$

It is then necessary to check if a specific combination of features exists in the dataset. Note that we do not have control over all the features when creating a time series. We only determine the initial Hurst value of the irregular component, the values of the coefficients, the combination model, the length of the time series, and measure the features on the final time series. The coverage for j^{th} category of feature f_i ($1 \leq j_i \leq n_i$) is indicated by $C(i, j_i)$ and calculated by Eq. 5.26.

$$\begin{cases} C(i, j_i) = 1 & \text{Combination exists} \\ C(i, j_i) = 0 & \text{otherwise} \end{cases} \tag{5.26}$$

Therefore, the Coverage Rate (CR) for the entire dataset is calculated by Eq. 5.27. Based on Eq. 5.27, CR is always between zero and one, or more precisely $0 < CR \leq 1$. The highest possible value for the coverage rate (CR) is 1, which indicates a full coverage. The lowest value for CR, however, is always greater than zero on a non-empty dataset, as a non-empty dataset always contains at least one combination.

$$CR = \frac{\sum_{i=1}^{M} \sum_{j_i=1}^{n_i} C(i, j_i)}{f_S} \tag{5.27}$$

For this evaluation, we selected a measure of diversity that reflects the *percentage coverage* of the samples over the *potential* feature space. Using Table 5.3, there are: 3 categories for spectral entropy; 3 categories for Kurtosis; 3 categories for Skewness; 2 categories for GoD; and 7 categories for DFA. Thus, based on Eq. 5.25, there is a total of ($3 \times 3 \times 3 \times 2 \times 7$) 378 possible feature combinations (f_S=378).

Fig. 5.18 shows the number of time series where a specific category was represented by our synthetic data while Fig. 5.19 shows the coverage of categories that the dataset represented.

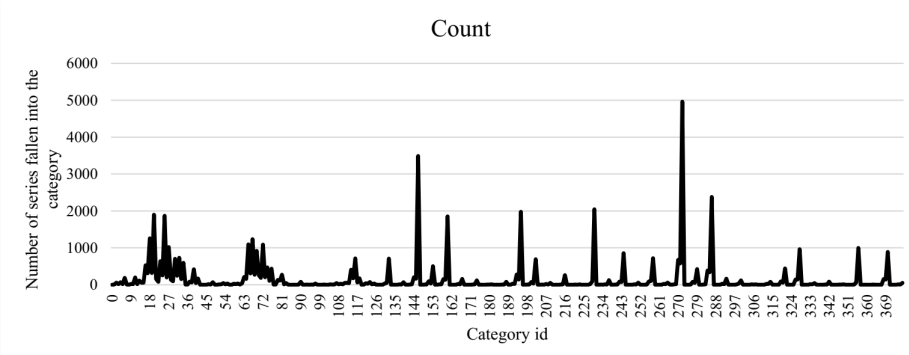


Figure 5.18: Number of series in each category

In Fig. 5.18, the horizontal axis entitled as "category id" represents 378 different categories of time series each corresponding with a unique combination of features, and the vertical axis represents the number of series attributed to the corresponding combination of features.

Based on Eq. 5.26 (as shown in Fig. 5.19), our 50K dataset has at least one time series for 272 combinations (out of the 378 possible combinations) and thus, based on Eq. 5.27, the coverage percentage for our dataset is 72%. This demonstrates that our dataset exhibits a significant level of coverage over the feature space and thus, is an appropriate benchmark for time series prediction algorithms assessment.

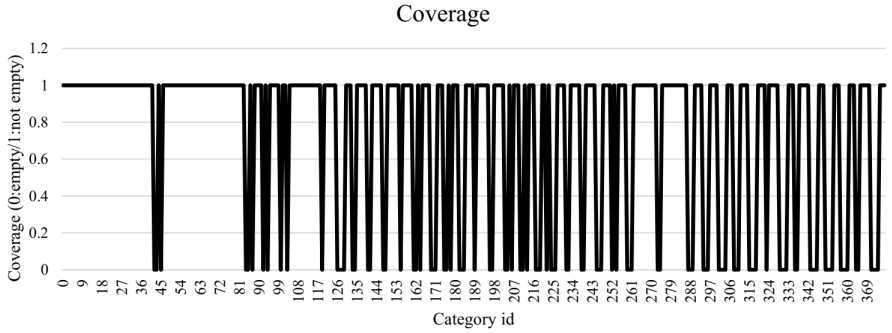


Figure 5.19: Coverage

Sensitivity Analysis

We also performed a sensitivity analysis on the categories of Spectral density and GoD, for which we did not provide references supporting our choice of categories. First, the threshold for GoD has been changed from 0.02 to 0.05, and thus $p(GoD < 0.05) = 0.71$ and $p(0.05 \leq GoD) = 0.28$. As a result, the Multi-variate Entropy Score = 0.834 and the coverage rate = 0.719. Second, the number of categories in Spectral entropy has been changed from 3 to 2 categories, and the threshold has been set to 1. Thus, $p(Spectral_Entropy < 1) = 0.425$ and $p(1 \leq Spectral_Entropy) = 0.575$. As a result, the Multi-variate Entropy Score = 0.830 and the coverage rate = 0.821. Note that Spectral entropy does not provide a standard (a value between 0 and 1) metric. In order to address this issue, we replaced Spectral entropy with SVD entropy (SVDEnt), with three categories: $A = SVDEnt < 0.2$, $B = 0.2 \leq SVDEnt < 0.8$, and $C = 0.8 \leq SVDEnt$ [241]. As a result, the Multi-variate Entropy Score = 0.779 and the coverage rate = 0.7248.

5.6 Conclusions

Researchers using time series data are often faced with the problem of insufficient data for the purposes of testing and validating their algorithms. In this Chapter, we presented a methodology for the creation of a large number (53,637) of time series which are made available to the research community [20]. Their construction had

an emphasis on *diversity* and a validation framework to ensure a robust evaluation of the synthetic datasets created. Our method comprised 5 well-known time series features and used a multivariate entropy measure to examine the diversity of the created time series based on these five features. The experimental results showed that our overall dataset measured diversity at 83.4%, which we believe to be a significant achievement. We have also proposed a new diversity assessment measure called the *coverage rate* which reflects the coverage of the dataset over the full feature space. The results show that our series exhibit a coverage rate of 72%, which delivers a significant contribution for such a large dataset.

Chapter 6

A Meta-Learner for MSAP Model Selection

In the final part of the 3-step approach to this research, we turn our attention to the requirement for a meta-learning process which attempts to select the best performing model from a set of candidate models. We begin with an introduction in section 6.1 where we discuss challenges and our contribution to tackling this particular problem. In section 6.2, we formalize a methodology that helps identify an appropriate time series prediction method based on a meta-analysis of the characteristics of that time series. In section 6.3, we implement our proposed strategy to provide a validation of its strengths and weaknesses. We finish this Chapter by providing some conclusions in section §6.4.

6.1 Introduction

With the increase in the number of available time series prediction methods, the complexity in time series model selection becomes more acute. Generally, the decision process required when choosing a method can involve a detailed assessment of the time series using metrics such as those outlined in Chapter 5. Even when

using 5 metrics as in our experiments, the number of possible combinations of the discretized time series characteristics rose to 378.

As discussed in Chapter 2, there seems to be two broad approaches to finding prediction solutions: parametric and non-parametric estimation methods. Parametric methods such as ARIMA have proven to be successful in the past. However, there is an increasing trend in the use of non-parametric machine learning approaches, which in theory make less assumptions about the distribution of the data [129]. In practice, machine learning approaches require a substantial amount of computation time and convergence has been found to be an issue, as there often exists a high level of non-linearity in the solution space [170].

Each time series dataset has a unique set of time series characteristics, and as stated in Chapter 2, over the complete time series feature space no one method outperforms all others. This phenomenon is known as the "no free lunch" theorem [272]. Traditionally, practitioners have used their own expert knowledge to relate time series to an appropriate model [222]. However, this solution is now becoming less appropriate with the growing number of models available and thus, requires an automated approach to identify candidate methods using metadata collected from a time series [156]. Thus, in this final part of our research, we propose a model selection approach that takes a set of time series features as inputs and estimates the performance of a given prediction method.

Finding the appropriate prediction method for a given dataset is a continuing challenge in time series prediction problems. Generally, it requires the researcher to match the time series characteristics with the most appropriate models. This field of study is known as *Meta-Learning* and requires the researcher to characterise a time series with a number of metrics, which are then used in the selection process of an appropriate algorithm. A meta-learner is typically developed using a classifier such as a Support Vector Machine (SVM) or a decision tree.

Time Series analysis has traditionally been implemented using parametric techniques, which have relied upon method specific assumptions being met. With the

advent of machine learning in time series analysis a considerable number of new approaches have become available to researchers. Theoretically, machine learning approaches have no predefined assumptions but in practice, it has been shown that methods like RNNs outperform NNs when there is long term memory in the time series. Additionally, if the solution space for the individual method is close to linear, then parametric methods should have comparable predictive performance to machine learning techniques [87]. In this research, we propose a Meta-Learning strategy that attempts to identify appropriate time series prediction methods from a list of well known and popular approaches. We use ARIMA, NN, SVR, LSTM and RNN as the candidate models. It should be noted that SVR, LSTM and RNN have not been included in previous Meta-Learning studies. In addition, the existing Meta-Learning studies have predominantly focused on one step ahead prediction strategies while the focus of our research is multi step ahead prediction. Finally, our approach also focuses to the problem of hyper-parameter selection which has not received significant attention in the literature.

Challenges.

While the use of meta-learning in times series is gaining traction, there are still challenges to be addressed. We highlighted in Chapter 2 that the majority of research efforts in meta-learning for prediction method selection, did not include machine learning techniques as candidate models but chose to focus on parametric methods. Furthermore, studies such as [156] which used machine learning techniques as candidate models, did not address the complexities arising in the training phase of these techniques. What was also shown in the literature review in Chapter 2, was that existing meta-learning strategies have not addressed issues caused by feature space coverage, hyper-parameter selection or model uncertainty caused by the random initialisation phase in machine learning prediction models.

In Chapter 2, we reviewed how previous approaches have not used state of art techniques, such as SVR, RNN and LSTM, as candidate models in their meta-learning strategies. While we intend to adopt these approaches, choosing hyper-

parameters and initial random assignments of the input weights will have an impact on the capacity and performance of the model and thus, can adversely affect the models rating when compared to other approaches. This will require the usage of bootstrapping to help alleviate the problems caused by hyper-parameter selection and model uncertainty.

Contribution.

This research proposes a Meta-Learning strategy that can be used to help identify an appropriate multi step ahead prediction (MSAP) approach, given a particular set of time series features as inputs. One important difference between our approach and the existing approaches is that unlike existing approaches which use classifiers in the meta-learner, we use a regression model to implement the meta-learning strategy. Using this regression model we estimate the performance of the given candidate models on a given time series. This will give the practitioner the flexibility to avoid the uncertainties involved in relying on a single preferred prediction model.

In Chapter 5, we presented a range of metrics that are currently used to interpret, understand or describe a time series dataset. These metrics have been used (partly) as input features in the training of the learning component for previous meta-learning learning strategies and a selection of them will be used in this research to implement meta-learning. In the validation phase of this part of our research, we show that these metrics such as DFA, Hurst Exponent, and Shannon entropy, can be used as inputs to identify the MSAP method with the lowest normalized Mean Square Error (nMSE), and incorporate two MSAP strategies in our analysis: REC and MRFA. Finally, we include an approach to manage hyper-parameter selection to address issues of poor performance.

6.2 Meta-Learning Methodology

The principle behind the meta-learning is based on identifying an appropriate multi step ahead prediction (MSAP) method given a set of characteristic metrics taken

from a specified dataset. Inputting these metrics into a pre-built model results in a recommendation. For this part of our research, we use a regression model to implement meta-learning. However, we may refer to the proposed model by different terms such as meta-learner or the prediction model selection approach.

Our approach is designed to model the performance behavior (error) of a prediction model with respect to a set of time series meta-features. This provides us with a system that has the generalization ability to estimate the performance of the prediction model on unseen time series. In effect, it will attempt to estimate the error of a given model with respect to the meta-features of a given series, and can be broken down with the following components as part of the overall architecture:

- Input Features: a set of time series features listed in section 6.2.1.
- Candidate Model Pool: the set of MSAP models outlined in section 6.2.2.
- Hyperparameters Settings: hyperparameters for each model as explained in section 6.2.3.
- Standardized error: the error metric that is independent of the magnitude of the signal, explained in section 6.2.4.
- Regression machine learning model: The regression model used for implementing the meta-learner which is described in section 6.2.5.

In our proposed approach, the goal of the meta-learner is to approximate a function denoted by $\mathcal{G}$. Formally, $\mathcal{G}$ calculates the *error* of the model m when used to predict a given time series with the set of time series meta-features F. The output of the meta-learner is described by Eq. 6.1, where $\mathcal{G}$ is the solution of model m on with features F.

$$error = \mathcal{G}(m, F) \tag{6.1}$$

6.2.1 Input Features

The input meta-features used in this research for implementing our Meta-Learner are listed below and fully described earlier in Chapter 5, where diversity was discussed in terms of time series characteristics.

- **Shannon Entropy:** For a signal y with sample size N, sample entropy is the negative logarithm of the conditional probability that a sub-series of length M matches point-wise with the next point with tolerance (distance less than) r.
- **Spectral entropy:** Spectral entropy is calculated using Shannon entropy and quantifies the spectral complexity or randomness of the power spectrum of the time series over a long period of time.
- **SVD entropy:** SVD entropy is an indicator of the dimensionality of the time series, i.e. the number of eigenvectors needed for an adequate description of the given time series.
- **Fisher information** (FI) is a measure of information content in data. FI quantifies the amount of information the data represents regarding unknown parameters and determines how much information can be obtained from a specific amount of data.
- **Kurtosis** measures the number of outliers in the dataset with respect to a normal distribution: when Kurtosis is high, the number of outliers is high; when kurtosis is low, the number of outliers is low or zero.
- **Skewness** measures the symmetry of the distribution.
- **Gaussianity of the Differences** (GoD) measures the normality of the distribution of the first lag difference of the time series. We implemented the Shapiro-wilk test.
- **The Hurst exponent** H attempts to explain LRD as a property of stochastic self-similar processes.

- **Detrended Fluctuation Analysis** (DFA) is a method for evaluating the statistical self-similarity of a signal.
- **Stationarity** can be defined as the state where the statistical properties of the given time series such as mean, variance and auto-correlation remain steady over time.

6.2.2 Candidate models

As stated previously, the objective of our meta-learning approach is to estimate the $nMSE$ based on a proposed multi-step ahead prediction **MSAP**. However, an MSAP approach consists of two components: the MSAP strategy (e.g., recursive *REC* or multi-resolution forecast aggregation *MRFA*), and the prediction model (e.g., neural network *NN*, recurrent neural network *RNN* and support vector regression *SVR*). The prediction model is the core computational element of an MSAP strategy and thus, an MSAP strategy (recursive *REC*, direct *DIR* and multi-resolution forecast aggregation *MRFA*) can be implemented using different prediction models. For instance, REC, DIR and MRFA (as MSAP strategies) can be implemented using either NN, RNN or SVR. Therefore, any MSAP approach is a (Strategy-model) $strategy^{model}$ combination like REC^{NN}, REC^{SVR}, REC^{RNN}, DIR^{NN} or $MRFA^{RNN}$. In this research, we refer to each $Strategy^{model}$ combination as a separate *candidate model*. We now provide a brief description of each model used in our experiments and validation.

MSAP Strategies. We previously reviewed existing MSAP strategies and discussed their strengths and limitations in Chapter 2. We learnt that except for the recursive strategy (REC), all others suffer from intermediate information loss due to discarding serial correlation. We developed MRFA based on the principles of the recursive strategy (REC) to improve the accuracy of MSAP. We now focus on REC and MRFA as the core MSAP strategies for the meta-learner, since they were developed to retain serial correlation when making predictions.

Prediction models. In Chapter 2, we reviewed the current state of the art time

series prediction models including ARIMA, SVR, NN, RNN, and LSTM. In this research, we are specifically interested in using machine learning approaches as candidate models due to their recent popularity and success in time series prediction application. We will first briefly study the use of these models in the past meta-learning studies (focusing on time series prediction). Table 6.1 investigates the use of the current state of the arts prediction models in the past meta-learning studies; wherein at least one machine learning prediction method was used in their set of candidate models.

Table 6.1: Candidate Models used in existing Meta-Learning Approaches

Works	**Year**	**ARIMA**	**SVR**	**NNs**	**RNNs**	**LSTM**
[262]	2009	✓	✗	✓	✗	✗
[212]	2010	✓	✗	✗	✗	✗
[156]	2010	✓	✗	✓	✓	✗
[215]	2011	✓	✗	✗	✗	✗
[269]	2013	✓	✗	✗	✗	✗
[89]	2014	✓	✗	✓	✗	✗
[147]	2016	✗	✗	✓	✗	✗
[238]	2016	✓	✗	✓	✗	✗

Table 6.1 shows that ARIMA has been the most popular candidate model, followed by NN. RNN was used once, in spite of its widespread popularity in time series analysis over the past decade. SVR and LSTM have not been studied as candidate models in the past meta-learning studies. However, LSTM has recently received considerable attention in time series prediction applications, primarily due to its ability to handle long term memory. Also, SVR has been widely used for prediction purposes in time series studies, because of its good generalization abilities. Our reasons for using ARIMA, NN, RNN, SVR and LSTM is described below, with Table 6.2 summarising the strategy-model combinations that are used as candidate models in this research.

ARIMA [47] has been a very popular prediction model in the time series community and has shown a powerful prediction ability among the stochastic prediction

Table 6.2: Strategy-Model Combinations for Candidate Models

Engine \ Strategy	**REC**	**MRFA**
ARIMA	REC^{ARIMA}	—
SVR	REC^{SVR}	$MRFA^{SVR}$
NN	REC^{NN}	$MRFA^{NN}$
RNN	REC^{RNN}	$MRFA^{RNN}$
LSTM	REC^{LSTM}	—

models. ARIMA pays a particular attention to the autocorrelation structure in the time series and thus is an important prediction model. ARIMA is a parametric model and is not able to capture the non-linear structure of time series. Differencing has been widely used to remove long term components such as the nonlinear trend-cycle component. However, when dealing with a nonlinear trend-cycle component, differencing can leave unknown non-linearities in the results. Since ARIMA is incapable of implementing forward smoothing, it cannot be used for modeling ROI and thus, ARIMA cannot be used as the model in MRFA. ARIMA was only implemented in the REC strategy (REC^{ARIMA}).

SVR [189] is gaining popularity due to its good generalization ability and its low complexity. We used SVR in our implementation due to the following: its ability in solving high-dimensional problems; SVR avoids local minima and overfitting problems; and it needs less a priori known parameters in comparison with ANN [233]. We used SVR to implement both REC and MRFA (REC^{SVR} and $MRFA^{SVR}$).

A **Neural Network** [73] was added to the candidate pool due to its widespread popularity in time series prediction community. The biggest challenge in implementing the NN models is the choice of hyper-parameters. We used NN with both REC and MRFA strategie: We have both REC^{NN} and $MRFA^{NN}$.

A **Recurrent Neural Network** [75] is a powerful machine learning technique for solving sequential problems. The incorporation of feedback recurrent weights in the structure of RNNs allows them to keep track of long historical mid-term results of

the network. We used RNN due to its ability in processing sequential data. RNN was used with both REC and MRFA strategies (REC^{RNN} and $MRFA^{RNN}$).

LSTMs [118] are increasing in popularity because of their ability to avoid the vanishing/exploding gradient in RNNs which allows LSTM to process longer term memories. We used LSTM to implement REC but due to the lack of sufficient computational resources, we could not use LSTM to implement MRFA: (we only have REC^{ARIMA})

6.2.3 Machine Learning Hyper-Parameter Tuning

In this research, we have a particular interest in using machine learning candidate models such as NN, SVR and RNN, which were originally developed to learn from the data and thus, are forced to address two major issues: 1-the choice of hyper-parameters and 2-Selection of the validation set.

Hyperparameters Selection. In machine learning problems, hyperparameters are the non-learned parameters which control the capacity of the model for learning the dynamics of the given problem. Hyperparameter optimization has been identified as one of the major challenges in the relevant literature. However, it is a time consuming and computationally expensive task and thus, hyperparameters optimization is impractical when a large volume of prediction experiments should be implemented. To build a meta-learner for prediction method selection, a large number of time series should be predicted by a number of candidate prediction models. Therefore, applying hyperparameters optimization to each candidate model for each individual time series is a computationally demanding and hence, impractical task.

Validation Set Selection. When training a machine learning prediction model, the given dataset is split into two sets: the training set and the validation set. The validation set is used to avoid overfitting and check if the learning objective is reached. Selection of the validation set is another task in machine learning problems that should be managed carefully in order to avoid overfitting.

In this approach, to resolve the problem of hyperparameter tuning for each prediction model, we propose the following strategy: for each hyper-parameter h_i ($1 \leq i \leq I$), a set of choices c_i is considered and the model's error (in terms of $nMSE$) is recorded for all the possible combinations of hyperparameters (a total of $\prod_{i=1}^{I} c_i$ combinations). At the end, the average $nMSE$ is reported.
However, for each individual combination of the hyper-parameters, we measure the error ($nMSE$) using a bootstrapping process in order to avoid overfitting. In a bootstrapping process, a number of k=5 validation sets are randomly selected from the data and the average prediction error over the selected set is reported. These validation sets are out of sample continuous bootstrapped samples.

Depending on the choice of hyperparameters, machine learning models can exhibit instability at the output, meaning that the model produces a different output at each experiment (given the same input). This instability is due to the incorporation of random initial weights (for instance in neural networks), which is a part of the learning algorithm. To overcome the output's instability, we repeat each experiment d=10 times and report the average error ($nMSE$).

Therefore, the total number of experiments is calculated by Eq. 6.2, where N is the number of the available candidate models, and I_j is the number of hyper-parameters in the j^{th} candidate model.

$$N^{Experiments} = d \times k \times \sum_{j=1}^{N} \prod_{i=1}^{I_j} c_i \tag{6.2}$$

Hyperparameters Variables. For each series, the order of the $ARIMA(p, d, q)$ model was obtained using the exact maximum likelihood using a Kalman filter. A detailed discussion on optimal ARIMA models is beyond the scope of this research and is expanded upon in detail in [279,291]. The ranges of the hyperparameters used in the evaluation of machine learning models this research are outlined as follows:

- The **Lags** or simply the size of the sliding window employed to predict the given time series, ranges from 4 to 12 steps backward.

- The set of neurons in the hidden layer of the NN model was {4,6,8,10,12}.
- The number of Neurons in the hidden layer of the RNN model was {4,6,8,10} and the number of recurrent connections was {1,2,3,4,5,6,7}.
- The C-value for the SVR model ranges was {0.1,0.5,1,10,100,1000}.
- The number of cells in the LSTM model was {6,8,10,12}.

Note that for seasonal data the lag might be 24 or 365. However, we assume that there is no prior knowledge about the size of seasonality. If the size of seasonality is known, the signal should be deseasonalized before making predictions.

6.2.4 Standardized Error

As stated previously, the initial goal of the meta-learner is to estimate the error of a candidate prediction model given a certain set of time series meta features. To implement our method, an error metric is needed to represent the prediction accuracy of candidate models that is independent of the magnitude of the signal. The Mean Squared Error (MSE) is the most popular metric of prediction accuracy in regressions models and is calculated based on the signal's magnitude, 6.3. However, this metric must be standardized in order to relate each machine learning methods performance, as it incorporates a scale dependency with the original series.

$$nMSE = \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - t_i}{Max(y) - Min(y)})^2 \tag{6.3}$$

This research makes use of the normalized MSE (nMSE) to overcome the problem of scale dependency in the MSE. The nMSE provides independence of scale, and is calculated using Eq. 6.3. In order to improve the performance of our Meta-Learner, we trained it using $\log(nMSE)$.

6.2.5 The Regression Model

In order to predict the candidate model error for a given time series, a machine learning regression approach needs to be implemented on the results of each candidate model (strategy-method combination) implementation for each series. The features of each series and a machine learning regression model (the meta-learner) will then be used to predict the nMSE value. The schematic view of our regression based Meta-Learner is shown in Fig. 6.1, where time series features and candidates from the Model Pool are fed into the regression model which generates a score for each combination.

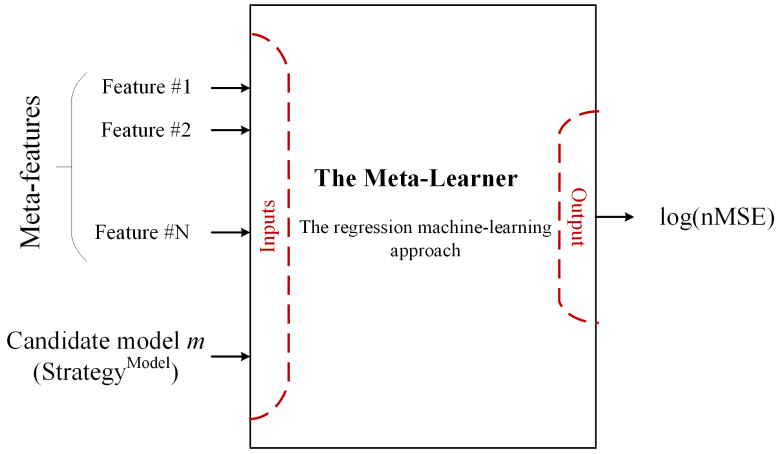


Figure 6.1: Schematic View of Regression based Meta-Learner

The Meta-Learner can be implemented using any machine learning regression model such as NN or SVR, that approximates a real-valued variable (nMSE of of a candidate model) given a set of features. However, to provide better accuracy, our approach used an Ensemble regression model. A lot of studies into ensemble methods have been carried out and they tend to assert that an ensemble regression approach (with combining multiple regression models) presents a better generalization ability than individual models [59, 161]. Ensemble approaches have shown an enhanced robustness to noise and stability in comparison to single regression techniques [59]. Thus, we developed the meta-learner using Random Forest regression which is a well-known ensemble technique in the research community [161]. The Random Forest Regressor is a bagging ensemble technique and works based on regression trees.

The idea behind bagging techniques is that combining multiple methods will improve the accuracy. The RF regression approach can be considered as a meta-estimator that integrates the results of multiple prediction methods to give more accurate results. The learning algorithm of The Random forest regression allows the regression tree components to grow to the maximum depth of the data using a combination of variables, and will not be pruned back.

Studies such as [201] argue that choosing an appropriate pruning method is a more influential factor than the variable selection metrics for determining the performance of tree based algorithms [201]. However, the authors in [50] showed that as the number of regression trees in the Random Forest increases, the generalization error always converges even without pruning the regression trees and overfitting is not a problem due to the Strong Law of Large Numbers [88].

A Random Forest (RF) regression approach receives an input vector $X = x_1, x_2, ..., x_p$, where p is the number of input features. The RF regression approach then creates K regression trees $T_1(x), T_2(x), ..., T_k(x)$ wherein each time all the regression trees estimate one specific actual value $\hat{Y_1} = T_1(X), \hat{Y_2} = T_2(X), ..., \hat{Y_m} = T_m(X)$. Finally, the RF regression approach returns an average of all the outputs from the regression trees as the final output, using Eq. 6.4.

$$Output_{RF}(X) = \frac{1}{k} \sum_{k=1}^{K} \hat{Y_k}(X). \tag{6.4}$$

As shown in Fig. 6.2, the Random Forest Meta-Learner has two sets of inputs: time series features and candidate prediction models. It means that our meta-learner is trained to estimate prediction error for multiple candidate models. In order to improve the performance of our (RF-based) Meta-Learner, the candidate models are fed to the Meta-Learner using a binary feature vector of size $J - 1$, where J indicates the number of available prediction models. The encoding of J candidate models into the binary feature vector is illustrated in Fig. 6.3.

As shown in Fig. 6.3, the binary feature vector can only contain one non-zero

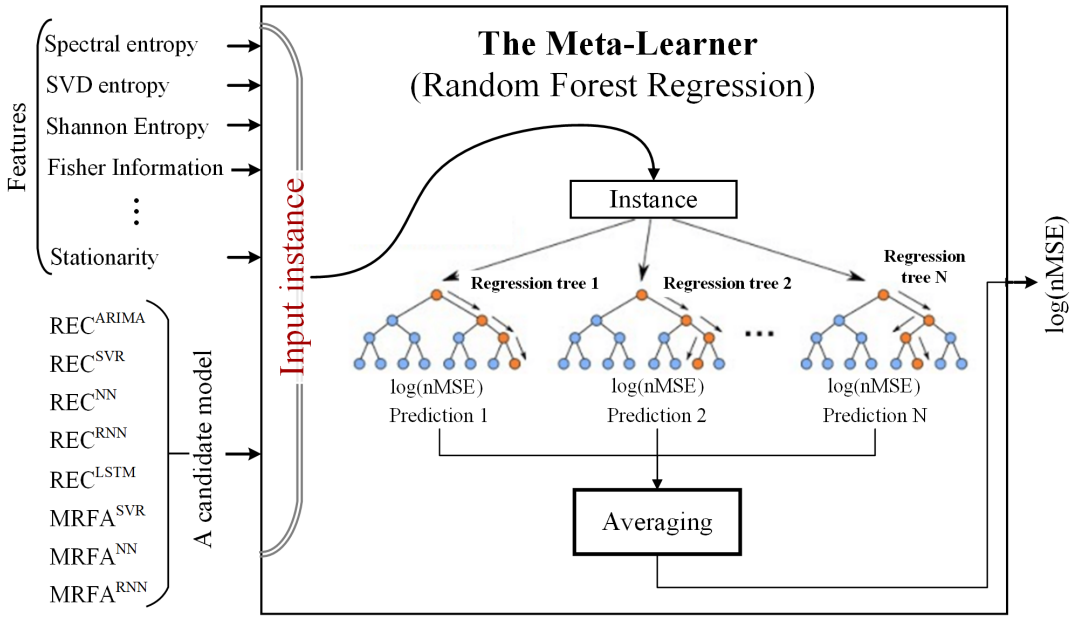


Figure 6.2: Random Forest Meta-Learner

element at a time. The first candidate model is represented by a vector of all zero elements, while the rest of the candidate models are each represented by one specific non-zero element in the feature vector.

7-Step Strategy. The steps to train the Meta-Learner and recommend the preferred method is outlines as follows:

- Extract features F from time series data.
- Apply candidate prediction models M to the time series and record $nMSE$.
- Train the Random Forest model using F and M as inputs, and $nMSE$ as the output.
- Extract features from the test data.
- Apply the model to the extracted features for each candidate model.
- Sort the candidate models based on their estimated nMSE.
- Recommend the model that has the least nMSE.

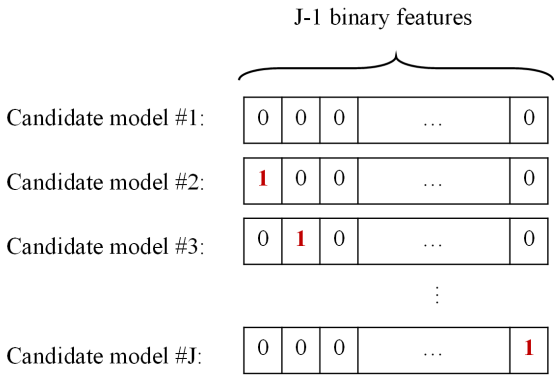


Figure 6.3: Encoding Candidate Models into a Binary Feature Vector

6.3 Evaluation

In this section, we present an evaluation of the meta-learner over a 20 step prediction horizon. We begin with a preliminary analysis in order to best configure the evaluation; then proceed to describing the experimental setup for our validation, before finishing with a presentation and discussion of the results.

6.3.1 Preliminary Analysis

As stated previously, the inputs to the meta-learner are the time series features and a candidate model, with the output being the model with the appropriate nMSE. This implies that the training dataset requires a candidate model ($strategy^{model}$) to be implemented on *each* time series. For our experiments, 8 candidate models were selected, and each of these required bootstrapping and a grid hyper-parameter combination strategy to be applied, giving rise to a large number of estimations for each series calculated by Eq. 6.2.

This is a considerable challenge and requires an extensive volume of computations for each series. In order to solve this, we decided to use the time series generation process outlined in Chapter 5 to create a synthetic dataset of time series. The experiments in this Chapter were run on a sample dataset containing 5,819 time series (out of the 53k series generated) due to hardware infrastructure limitations

on this project. As discussed in the previous Chapter, understanding the diversity of the dataset and its relevance to an overall feature space was measured using feature space coverage and diversity (multivariate entropy). In the dataset generated for this part of our research, we had a feature space coverage of 30 % and a diversity of 50%.

In order to determine that the dataset was well balanced, an experiment was conducted which compared a variety of MSAP model/strategy combinations. These included MRFA (using RNNs as the core prediction model) and REC (using ARIMA, NN, SVR, RNN and LSTM) to understand if there was a bias in the method choice for the dataset. Fig. 6.4 compares the performance of each MSAP method on the generated series. Each method is ranked in terms of its performance with the count of top ranked and second placed methods shown.

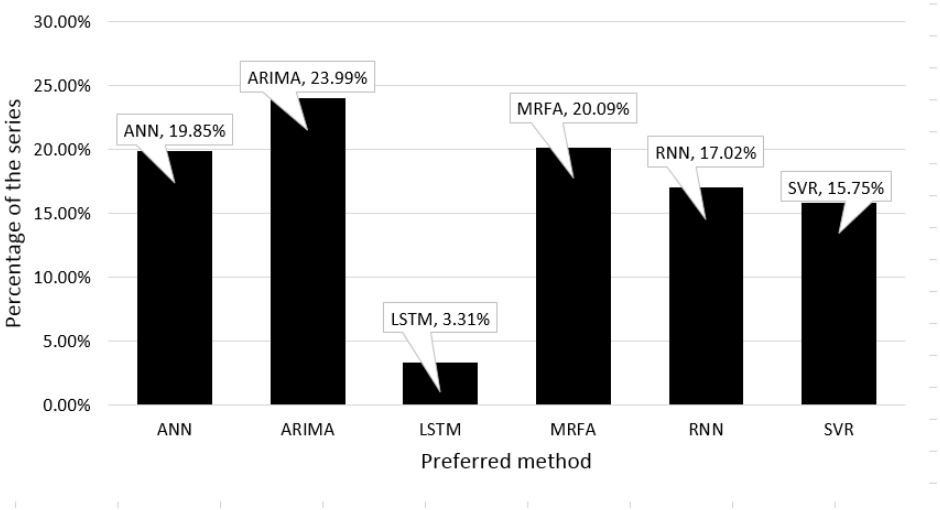


Figure 6.4: Performance Analysis

The results in Fig. 6.4 illustrate that the best performing model, ARIMA, performed best on only 24% of the datasets, with the MRFA model having a similar performance, was best on 20% of the datasets. Significantly, the results presented in Fig. 6.4 demonstrate that no single method outperformed the others across all datasets.

6.3.2 Experimental Setup

For the preliminary analysis, we implemented the recursive strategy (REC) using ARIMA, NN, RNN, SVR and LSTM and implemented the MRFA strategy using only an RNN. For the main evaluation, we implement MRFA using NN, RNN and SVR to further assess the performance of MRFA as a multi-step ahead prediction strategy. In fact, we implemented all candidate models introduced in Table 6.1. Note that, as stated in Chapter 4, MRFA is originally developed with RNN and this section implements $MRFA^{NN}$ and $MRFA^{SVR}$ for comparison purposes.

The results of applying the 8 candidate models (introduced in Table 6.1) to our series highlighted a number of series where $nMSE > 1$. Typically, the nMSE would be expected to be between 0 and 1. However, on certain occasions it can exceed 1 when predictions become highly erratic, and is an indication of poor performance or failure. Based on Eq. 6.3, if the difference between a prediction and its equivalent target value is greater than the difference between the maximum and the minimum values of the actual time series, i.e., $\hat{y}_t - y_t > max(Y) - min(Y)$ (where Y is the time series, y_t is the actual value at time t and $\hat{y}_t$ is the predicted value at time t), then $nMSE > 1$. The majority of the failures was caused by the incorporation of Brownian motion when simulating irregularity. This adds a high level of randomness to the time series leading to high errors. However, it also reflects poor poor prediction performance and is due to the inappropriate choice of model. For 8 steps prediction, a total of 7450 failures were identified, which is about 21.3% of the entire dataset (34914 records). In this analysis, failures were excluded as they can bias any proposed learner.

To investigate how successfully the *goodness* of fit for the performance estimator, our evaluation used the Pearson's correlation R as the main performance metric [31].

Note that our model is a meta-learning model whose output in the predicted nMSE. R was used to give a single metric for the performance of the model predictability of the nMSE. The output of R is a value between -1 and 1, where the proximity to 1 implies that the proposed model is demonstrating a reasonable fit. R is calculated

by Eq. 6.5:

$$R = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{6.5}$$

Having built the dataset, the Random Forest regression (ensemble) approach was used as the main approach to build the Meta-Learner and the following machine learning approaches were also used to compare the results: NN, SVR, and Decision Tree. We previously explained the typical architecture of NN and SVR approaches in Chapter 2, and explained the basis of a typical Random Forest regression approach in §6.2.5.

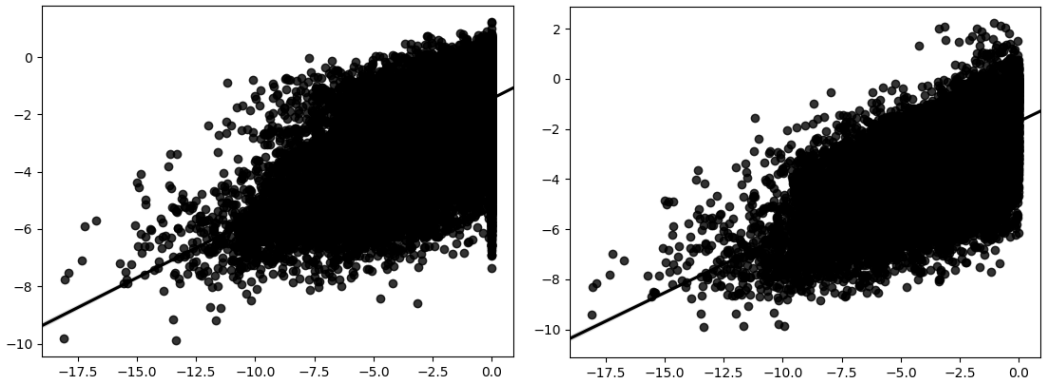
We implemented the NN model using a Multi-Layer Perceptron with one hidden layer and nine neurons in the hidden layer, which was the architecture that yielded the highest accuracy. The SVR Meta-Learner was implemented using a built-in grid-search python library called the skit-learn [207] for optimizing C and γ as the hyper-parameters of the model. We have already described SVR, Random Forest and Neural Networks. In this Chapter, we also use Decision Tree Regression to implements meta-learner and compared the results with our Random Forest regression model.

6.3.3 Results and Discussion

The initial implementation of the meta-learner adopted the R of the training and test data without excluding the failures for one step ahead prediction and is shown in Fig. 6.6 and Fig. 6.5, for the test and training data, respectively. We used a Multi-Layer Perceptron with one hidden layer and nine neurons was trained on 80% of the records, and the remaining 20% was used to validate the trained model.

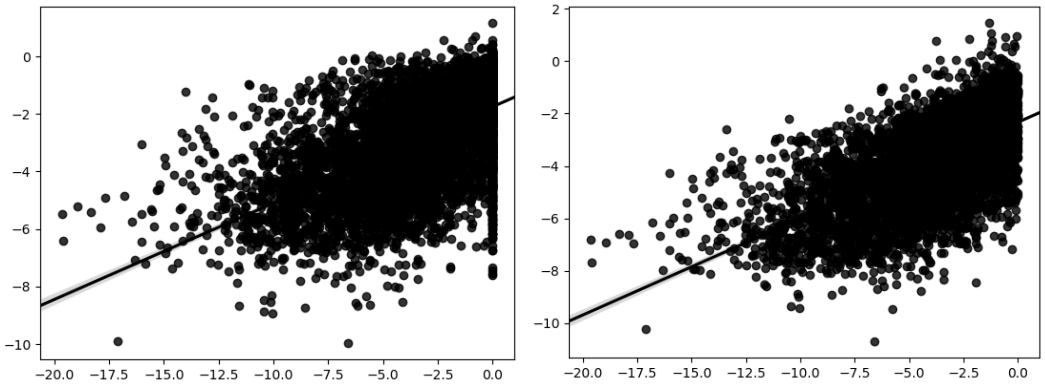
The results in Fig. 6.5 show that if the failures are not removed from the dataset, there will be a training bias (shown in Fig. 6.5a) on the estimation of $\log nMSE$ resulting that a broad range of values (from -7 to 0.8) are falsely estimated as zero.

This is crucial because it downgrades the performance of the meta-learner and leads to inaccurate recommendations. As shown in Fig. 6.5, the training bias has led to a significant number of wrong estimations (pay attention to the vertical line of points on the right side of Fig 6.5b). This demonstrates that the exclusion of the failures should be considered as a pre-processing step prior to the training of the meta-learner.



(a) Before failure exclusion (b) After failure exclusion

Figure 6.5: Actual versus Predicted values for training data before and after failure exclusion



(a) Before failure exclusion (b) After failure exclusion

Figure 6.6: Actual versus Predicted values for test data before and after failure exclusion

The results *before* and *after* excluding the failures from the test samples for one step ahead prediction are shown in Fig. 6.6.

Here, it can be seen that the removal of the failures from the training samples has resolved the issue of the bias. Thus, this decision resulted in better actual versus predicted figures, when compared with the results *before* excluding the failures.

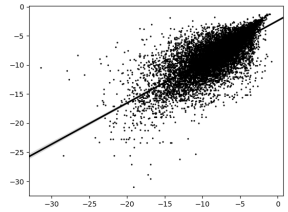
Comparing the results in Fig. 6.5b and Fig. 6.5b also illustrates that the removal of failures has led to narrower spreads around the regression line, both for training and test data and thus. enhanced the performance of the meta-learner.

The results are summarized in Table 6.3, showing that filtering failures leads to a significant improvement in the R of the performance estimator. We use the Random Forest Ensemble regression model as our proposed approach to implement the meta-learner.

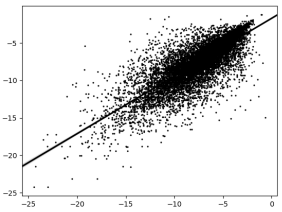
Table 6.3: R for the performance estimator

Data	R (unfiltered)	R (filtered)
Train	0.814	0.934
Test	0.802	0.916

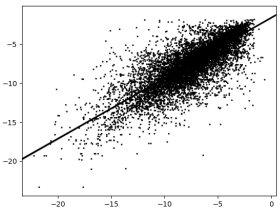
Fig. 6.7 shows the performance of our Random Forest Meta-Learner in terms of actual versus predicted values for 9 prediction steps.



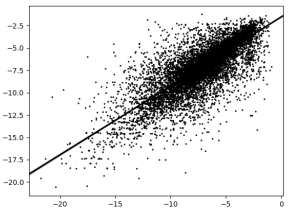
(a) One step ahead prediction



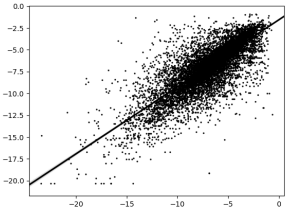
(b) Two step ahead prediction



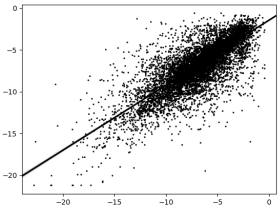
(c) Three step ahead prediction



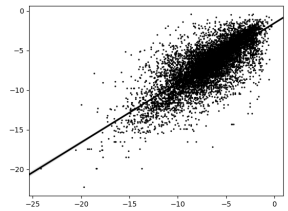
(d) Four step ahead prediction



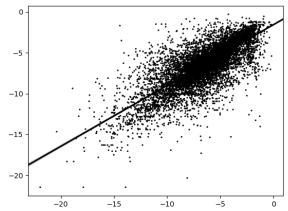
(e) Five step ahead prediction



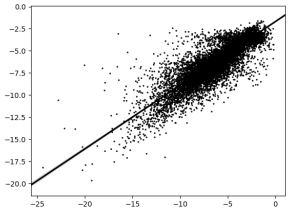
(f) Six step ahead prediction



(g) Seven step ahead prediction



(h) Eight step ahead prediction



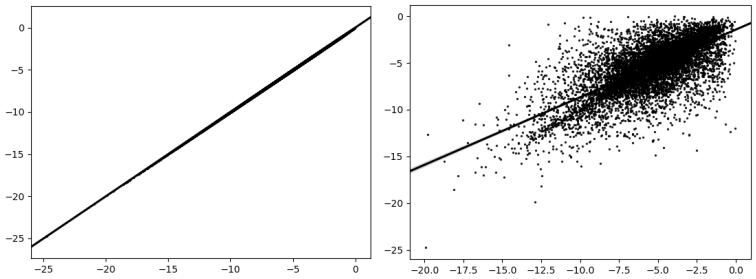
(i) Nine step ahead prediction

Figure 6.7: The actual versus predicted values for Random-Forest Meta-Learner over 9 Prediction Steps

These figures represent the actual $\log nMSE$ versus the predicted $\log nMSE$ and the closer the regression line to the identity line, the more accurate the meta-learner is. Also, the results show that the meta-learner performs better on higher steps ahead prediction. Note that the meta-learner estimated the performance of a prediction model and we use of the log function at the output of the meta-learner. The absolute value of the log nMSE is larger for more accurate predictions (at lower steps) and thus the difference between the predicted and actual value of nMSE is higher. Therefore, the meta-learner is more accurate in longer prediction horizons.

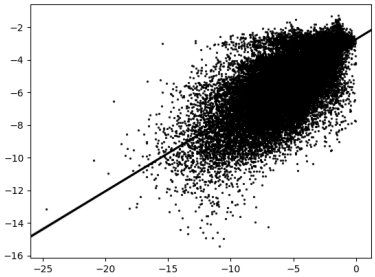
Regression Model Choice.

In this research, we use different methods to implement meta-learning to compare their performance with our chosen RF Meta-Learner. We implemented meta-learners using Neural Networks, SVR and Regression Decision Tree (DT). We used trial and error to obtain the best configuration of the used techniques as a more formalised approach is outside the scope of this research. Fig. 6.8 compares the performances of RF, DT, NN and SVR when employed to implement the Meta-Learner for 20 steps ahead prediction, for both the training and the test samples.

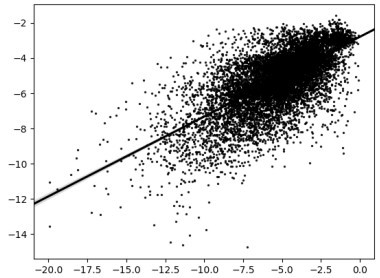


(a) Decision Tree ensemble regression train

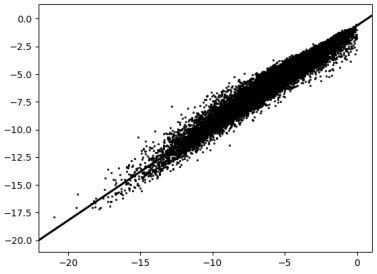
(b) Decision Tree ensemble regression test



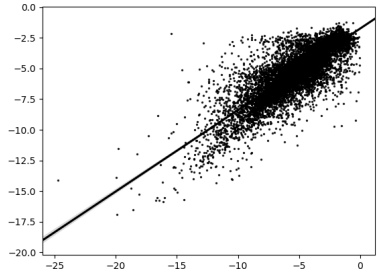
(c) Neural Network train



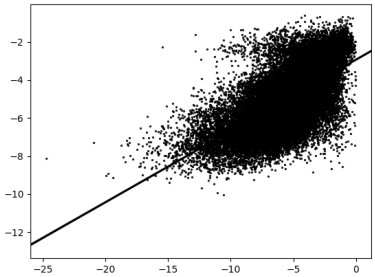
(d) Neural Network test



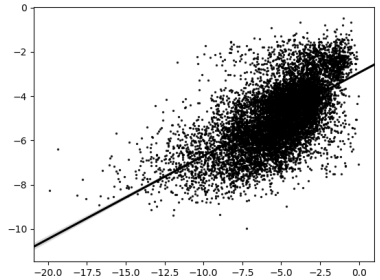
(e) Random Forest Ensemble train



(f) Random Forest Ensemble test



(g) SVR train



(h) SVR test

Figure 6.8: Comparing SVR, NN, DT and RF Meta-Learners in terms of predicted versus actual results

The left side of Fig. 6.8a shows the training performances of the meta-learners and plays an important role in our validation strategy, and demonstrates that over-fitting is not an issue.

Fig. 6.8a shows that, apparently, the decision tree model has overfitted to the training data and thus, is not a reliable choice for implementing the meta learner. Overfitting means that the model has overly fitted to the training data and thus, the model has only learnt to reproduce the observed data and fails on unseen data. Overfitting is not always a negative property, as many models may overfit but still perform well on out of sample test data; In this case unseen data are not far different from the training data and thus diversity is not important. This contradicts the purpose of our meta-learner where a generalization ability is required to predict $\log nMSE$ for unseen data [153]. Both Neural networks and SVR have shown a wide spread of the training points and their estimations around the regression line which indicates a their low performance when used for implementing the Meta-Learner.

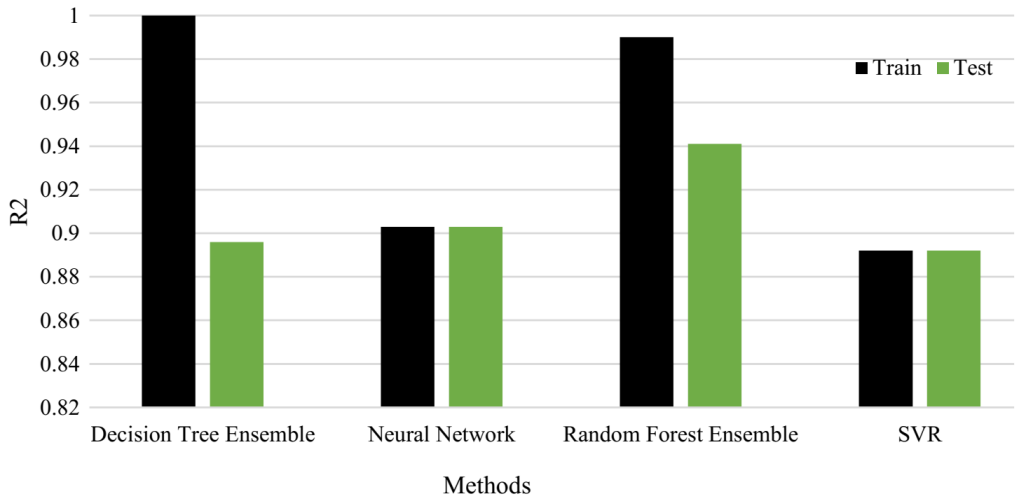


Figure 6.9: Comparison between Regression Models used in the Meta-Learner (R)

Fig. 6.9, compares the performances of the all four techniques used to implement the meta-learner for eight step ahead prediction. It can be seen that The decision tree has shown an R equal to one which is an indication of overfitting. Equal performances on the training and the test sets indicate that the test samples are very similar to the training samples and the model has only learnt to reproduce the trained samples. Both SVR and NN show similar performance for their train

and test sets which can be an indication of overfitting. The Fig. shows that RF exhibits a better performance than SVR and NN both for training and test samples; also, there is a significant difference between the R results of the training and test samples, which reduces the potential chance of overfitting.

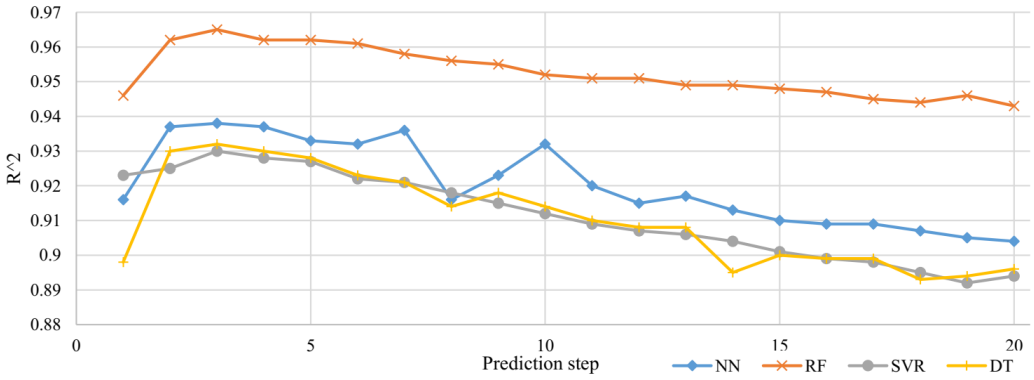


Figure 6.10: Comparison between regression models for implementing the Meta-Learner (R) over the entire prediction horizon

Fig. 6.10 compares the performances of the four machine learning techniques for implementing the Meta-Learner over the entire prediction horizon. It can be seen that for all the techniques used for implementing the Meta-Learner, the performance degrades as stepping towards the end of the prediction horizon. Fig. 6.10 also demonstrates the superiority of the RF model over the three other techniques used for implementing the meta model.

Evaluation on real time series

In order to evaluate the meta-learner on real data, the meta-learner was implemented on the 20 real time series introduced in Chapter 4. Based on the categorization strategy presented in Chapter 5, these 20 series cover 13 categories (out of 378) in the feature space. Also, the training set used to build the meta-learner covers 113 categories in the feature space. However, the 20 real series and the meta-learner's training set only share 7 categories. We split the 20 real series into two categories: 1-**Covered:** the series that their feature space is covered in the training data, and 2-**Non-covered:** the series that their feature space is not covered in the training data. There were 9 time series in the covered category and 11 time series in the non-

covered category. As mentioned earlier, there are 8 different candidate $Strategy^{model}$ models and thus, there will be $20 \times 8 = 160$ test samples for the meta-learner, in total. The distributions of the meta-learner's errors i.e., $Actual \log(nMSE) - Predicted \log(nMSE)$, for *covered* and *non-covered* series are compared in Fig. 6.11.

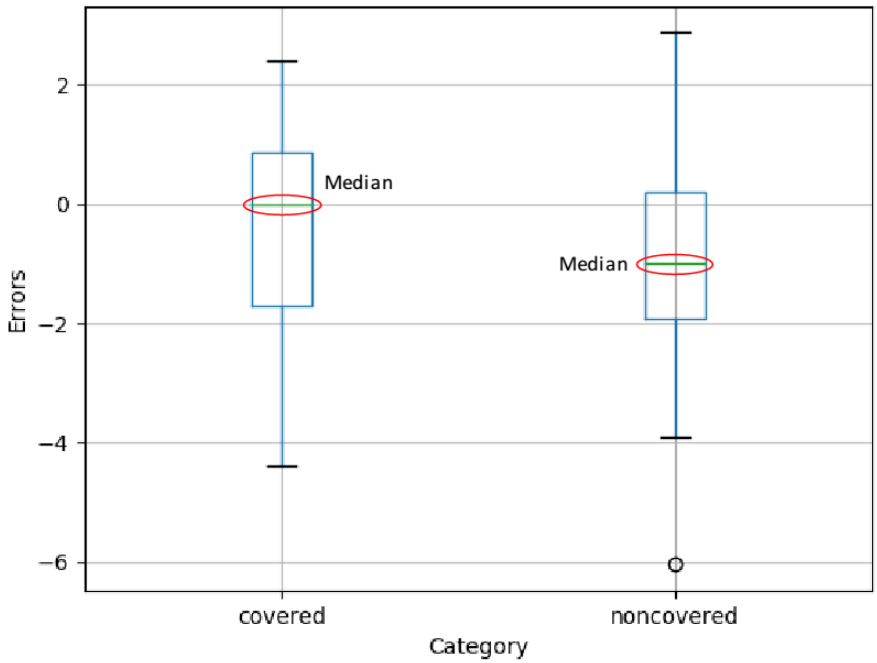


Figure 6.11: Comparing the results for the series covered and not covered in the training data

The results in Fig. 6.11 show that the median error for the *covered* series is zero, while the median error for the non-covered category deviates from zero, and this suggest that the meta-learner demonstrates a better predictive power on the series whose feature space is covered by the training data. This indicates that the Meta-learner actually reduces the error and thus improves the predictive power, when trained by appropriate data.

Fig. 6.12 illustrates our variance analysis on the zero-centred errors for covered and non-covered series and shows that the meta-learner has a lower error variance over the covered series.

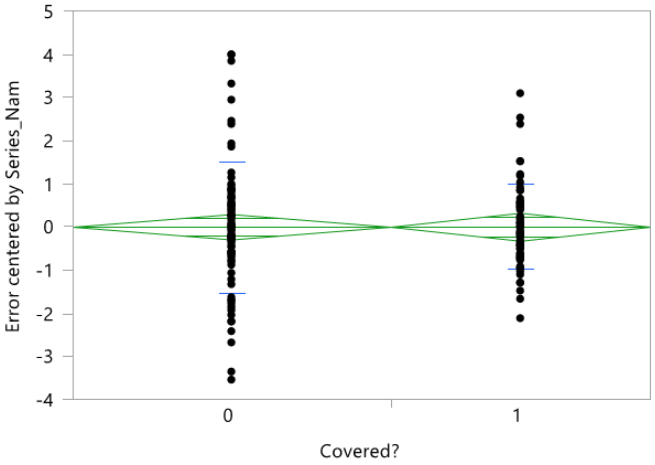


Figure 6.12: One way analysis of centred errors

We also performed a *t-test* on the centered errors, and the obtained standard deviation for the covered and non-covered series were 0.97 and 1.52, respectively. A one way ANOVA was performed on the 20 series using O'Brien test [198], Brown-Forsythe test [53], Levene [157] and 2-sided F-test. [32]. These tests are popular methods to test the assumption of the homogeneity of variances and were conducted in this research to compare the homogeneity of variances for the covered and non-covered categories. The results are summarized in Table 6.4:

Table 6.4: One way ANOVA on the centered errors

Test	**F Ratio**	**DF**	**p-Value**
O'Brien	6.6264	1	0.0111
Brown-Forsythe	5.9521	1	0.0160
Levene	5.8510	1	0.0169
F Test 2-sided	2.4400	1	0.0004

The results are summarized in Table 6.4 demonstrate that the hypothesis of homogeneity of variances is not supported and thus, the variances are different for the covered and the non-covered categories.

Evaluation of Ranks

The final goal of the meta-learner is to provide a ranking of the candidate models

from the most to the least accurate. We will then pick the top rank (the most accurate) models and introduce them to the practitioner as the recommended models. Note that the predicted ranks are obtained by sorting the candidate models according to the predicted $\log nMSE$ results (at the output of the meta-learner). The performance of the meta-learner, in terms of ranking accuracy, is evaluated by comparing the predicted ranks with the ranks induced by the actual $\log nMSE$. We performed a variance analysis on the differences between the actual and the predicted ranks and compared the results for the covered and non-covered series in Fig. 6.13.

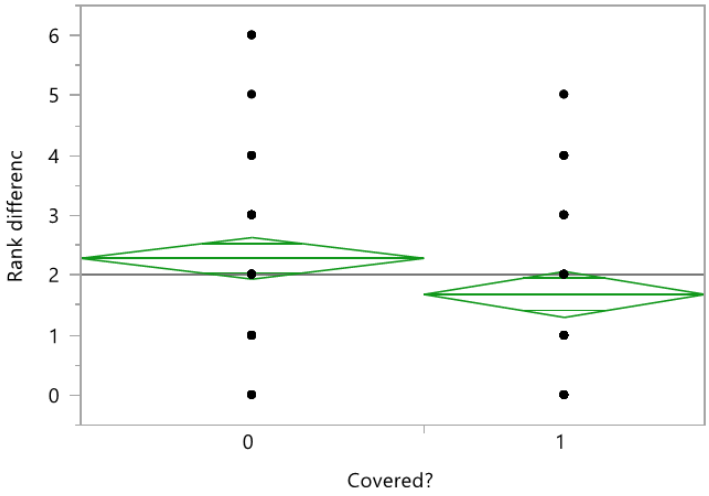


Figure 6.13: One way analysis of rank difference

As shown in Fig. 6.13, the variance of rank differences is significantly smaller for the *Covered* category and thus, the meta-learner presents a better ranking of the candidate models for the series whose feature space is covered in the training data. We performed a *t-test* assuming equal variances to test whether the mean of rank differences is different for the *covered* and the *non-covered* series. The *t-statistic* was -2.305 with 138 degrees of freedom, and a significance or p-value of 0.0226, which was well within the 95% confidence level.

To further investigate this issue, we also performed the Wilcoxon (Rank sums) tests on the ranks where the score mean for the covered and and non-covered series were

77.48 and 61.96, respectively. The Wilcoxon test indicated that the predicted ranks were statistically significantly more accurate for the *covered* category compared to the *non-covered* category, $Z = -2.294$ and $p - Value = 0.0217$.

The final step in the evaluation of the meta-learner is to investigate how accurately the meta-learner identifies the top rank models. The results on the 20 series show that, the meta-learner correctly identified the top two models for 7 out of 9 series from the covered category, and 5 out of 11 series from the non-covered category. This indicates that the meta-learner has 78% accuracy on the series that their feature space was covered in the training data and 45% accuracy on the non-covered series.

6.4 Summary

In this Chapter, we presented a meta-learning approach for selecting the appropriate method for univariate time series prediction using a set of time series features to recommend an MSAP model. Unlike existing approaches that use classifiers like [89, 147, 213, 215, 238, 269], we incorporated a regression method that estimates the log-based Mean Squared Error, nMSE. To train our meta-learner, we used the dataset that we previously introduced in Chapter 5, which was both sufficiently diverse and large to provide a solid generalization ability for our meta-learning approach. The R of the resulted predictions indicate that our meta-learner has 94% accuracy which we feel is a good result given the size of the time series dataset used.

Chapter 7

Meta-Learning: Findings and Discussion

In this thesis, we introduced a new MSAP strategy known as MRFA; a time series generation process with a new evaluation metric; and a meta-learner that uses a log based mean squared error to select appropriate models for any given time series. Our final task to is try to combine all of these research developments together to deliver some new findings. While our meta-learner is designed to select the best from a set of predictive models for a time series dataset, our first idea was to examine the benefit of the alternative approach: early elimination of those models that will never be appropriate for the time series under study. In section 7.1, we present an early detection system for predictive models not suited to the time series under study by researchers. Our second idea was to use the large synthetic time series generated in Chapter 5, together with the evaluation methodology developed in Chapter 6 to provide a tougher threshold for our MRFA predictive model. In section 7.2, this new evaluation is presented before we summarize the Chapter in section 7.3.

7.1 Early Detection of Inappropriate Models

During the development of the meta-learner, we implemented 8 different model types for each time series dataset. As stated in previous Chapter, each candidate model had an extensive bootstrapping and grid search applied. In a number of these experiments, we were unable to get reliable results and found the $nMSE > 1$. This indicates failure in delivering valid predictions, as the error has gone beyond the scale of the fluctuations in the original signal Y, i.e, $error > Max(Y) - Min(Y)$. An obvious assumption for $nMSE > 1$ is that the underlying candidate model is not an appropriate method for the solution space in hand. This can be caused by the functional form of the cost function and the level of complexity introduced by the interaction between the model and the dataset. However, this result is not unusual and in fact, should be expected as in many practical situations, we often find that certain machine learning approaches are unsuitable for specific types of data. Understanding which approaches are likely to fail should be highly beneficial as practitioners in the machine learning community must often try a number of methods before they come across one that works. In other words, they are likely to be many failed experiments before a successful model/dataset pairing is found. This is one of the disadvantage with non-parametric methods, as they have no underlying assumptions to guide the practitioner in their choice. This observation led us to consider where our meta-learner could be employed as an early detection system for inappropriate model choices.

Note that one reason for $nMSE > 1$ can be due to a convergence problem in the machine learning model's training process. The convergence problem may occur as a result of the small sample size problem, an inappropriate configuration of hyper-parameters or the failure of the learning algorithm. A detailed discussion of learning algorithms is out of the scope of this thesis. However, to ensure that convergence is not the reason behind failures ($nMSE > 1$), we used bootstrapping and also reported the results as an average of multiple repeats of the experiment each with a different settings of hyper-parameters.

7.1.1 Primary model

When considering an early detection (ED) function, the goal is to simulate a Binary function denoted by $\mathcal{ED}$. Formally, $\mathcal{ED}$ predicts $class = Failure$ or $class = Pass$ and is a function of the model m and the set of time series features F. The $\mathcal{ED}$ (early detection) meta-learner receives the same input parameters as the meta-learner introduced in the previous Chapter in section 6.2. However, the output (out) of this model is defined as a binary variable, Eq. 7.1.

$$out = \begin{cases} 0 & ; nMSE \leq 1 \\ 1 & ; nMSE > 1 \end{cases} \tag{7.1}$$

As the outcome of this approach is a binary variable, classification methods as opposed to the regression approaches used in the previous Chapter were deemed appropriate. This is described by Eq. 7.2, where $\mathcal{ED}$ is the solution of model m with features F. Here, we used the Random Forest (RF) classifier to implement the ED Meta-Learner and thus, $\mathcal{ED}$ in Eq. 7.2 is an RF classifier in our approach.

$$class = \mathcal{ED}(m, F) \tag{7.2}$$

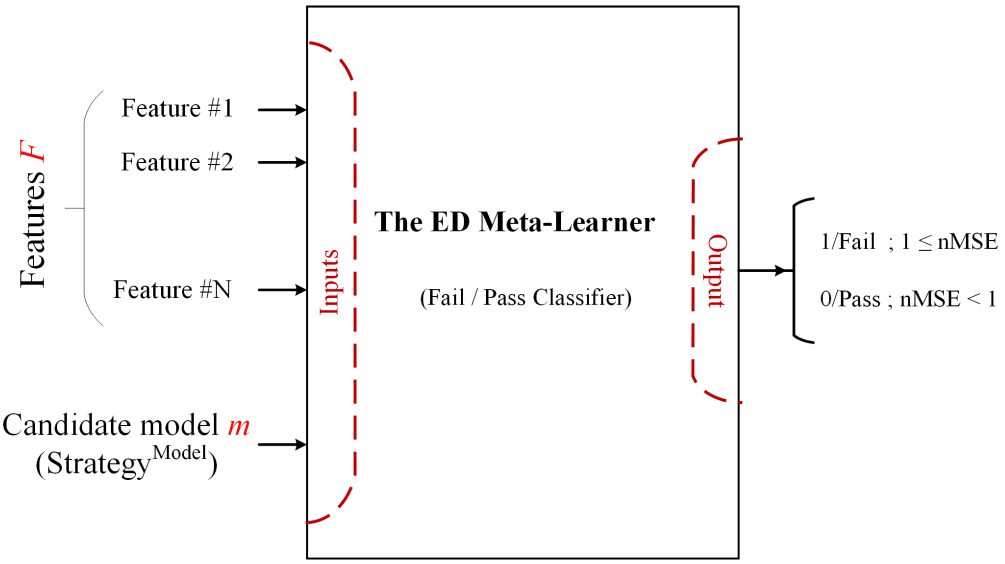


Figure 7.1: Architecture of the Early Detection (ED) meta-learner

Fig. 7.1 illustrates the architecture of the Early Detection (ED) meta-learner. Random Forest is a type of ensemble learning algorithm built upon the premise that multiple classifiers perform better that an individual classifier [77]. RFs are composed of many decision trees each making independent decisions, where decisions are then aggregated to optimize decision making. The Random Forest classifier is illustrated in Fig. 7.2. An important property of Ensemble classifiers is that they are more robust to outlying cases than individual classifiers.

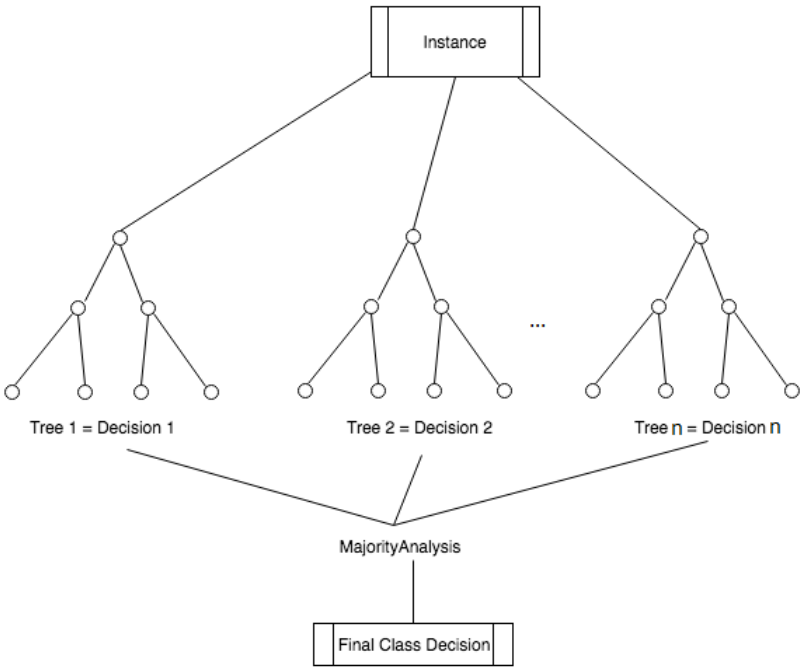


Figure 7.2: Random Forest Classifier

In the experiments for this analysis, an 80/20 split of the data was used for the train/test strategy. The performance of a classifier can be evaluated using a concept known as the confusion matrix, a table describing the performance of the classifier on a set of test data for which the true and false values are already known. In the confusion matrix, the following terms play the roles: **Positive (P)**: The sample is positive (the sample was actually a *Fail*) and **Negative (N)**: sample is not positive (the sample was actually a *Pass*);

- **True Positive (TP)**: sample is positive, and is predicted to be positive;
- **False Negative (FN)**: sample is positive, but is predicted negative;
- **True Negative (TN)**: sample is negative and is predicted as negative;
- **False Positive (FP)**: sample is negative but is predicted positive.

Note that because we present a failure detection model, a detection of a failure indicates a positive result.

Table 7.1: The performance of the failure classifier

		Predicted Class	
		Fail	Pass
Actual class	Fail	TP = 590	FN = 871
	Pass	FP = 361	TN = 5161

The results in Table 7.1 show that the overall accuracy is 82.36%, 590 records were correctly identified as failure ($TP = 590$), 361 records were incorrectly identified as failure ($FP = 361$), 5,161 records were correctly identified as pass ($TN = 5161$), and 871 records were incorrectly identified as Pass ($FN = 871$). The results suggest that when the output of the model is *Fail* (precision or positive predictive value), there is $\frac{TP}{TP+FP} \times 100 = 62\%$ chance that the output is correct, and when the output is *Pass* (negative predictive value) there is a $\frac{TN}{TN+FN} \times 100 = 85.56\%$ chance that the output is correct. These results can be visualized using a Receiver Operating Characteristic Curve (ROC), which is a graphical representation of *Sensitivity* or True Positive Rate (TPR) versus *specificity* or False Positive Rate (FPR) that indicates the discriminatory accuracy of a classifier as its decision criterion varies. In comparison with sensitivity and specificity, the Area under the ROC curve (AUC) is a preferable accuracy metric [199].

The ROC curve and the corresponding area under the curve (AUC) for the results of the RF classifier is shown in Fig. 7.3. AUC is the integral of the values represented by the black line in Fig. 7.3.

The results look positive on an initial examination of the RF classifier on the ED Meta-Learner, as the total accuracy of 82.36% is high and the AUC of 80.32% is also a good result. However, as shown in Table 7.1, there is a significant difference between the number of samples in the classes and this suggests we may have imbalanced classification problem. The *Pass* class is the majority and the *Fail* class is the minority and thus, the accuracy of the classifier will be biased towards the majority class. In the next section, we will present a solution to deal with the imbalanced

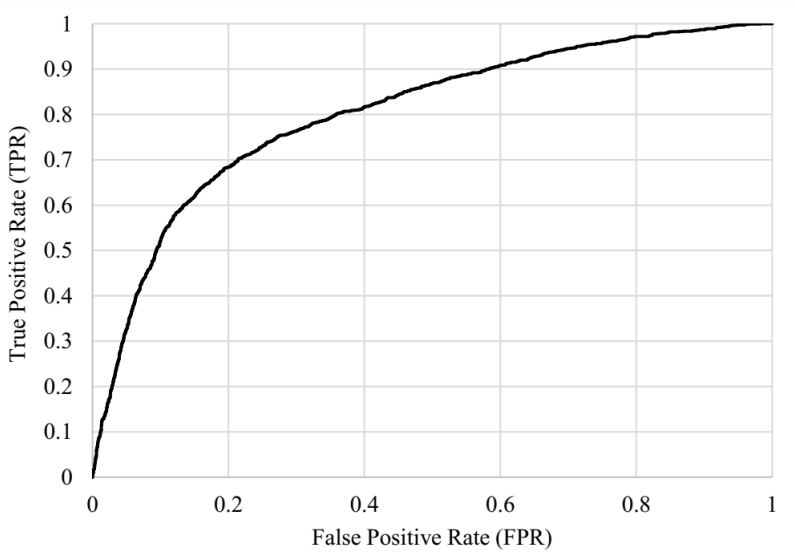


Figure 7.3: The ROC curve for the classifier

classification problem.

7.1.2 Resolving Class Imbalance

In this research, we use a technique known as Synthetic Minority Oversampling TEchnique (SMOTE) to overcome the class imbalance problem [58]. SMOTE suggests that the class imbalance problem can be tackled by oversampling the minority class. The algorithm first selects a sample from the minority class and identifies its k nearest neighbors. Then, a random sample from the identified neighbors is chosen and connected to the first sample to create the new synthetic sample.

Table 7.2: The performance of RF after SMOTE

		Predicted Class	
		Fail	Pass
Actual class	Fail	TP = 927	FN = 592
	Pass	FP = 172	TN = 1347

As shown in Table 7.2, the number of test samples has been reduced as a result of under sampling the majority class so that we have almost the same number of

samples in the majority (Pass) and the minority (Fail) classes. We can see that for positive predictions, $\frac{TP}{TP+FP} \times 100 = 84\%$ which represents a 22% improvement over the results before applying SMOTE. The negative predictive value is $\frac{TN}{TN+FN} \times 100 = 69.5\%$ which has been reduced but it no longer suffers from the class imbalance problem. Our original goal was to also achieve a good performance for the minority class and this improvement helps us to better detect inappropriate models.

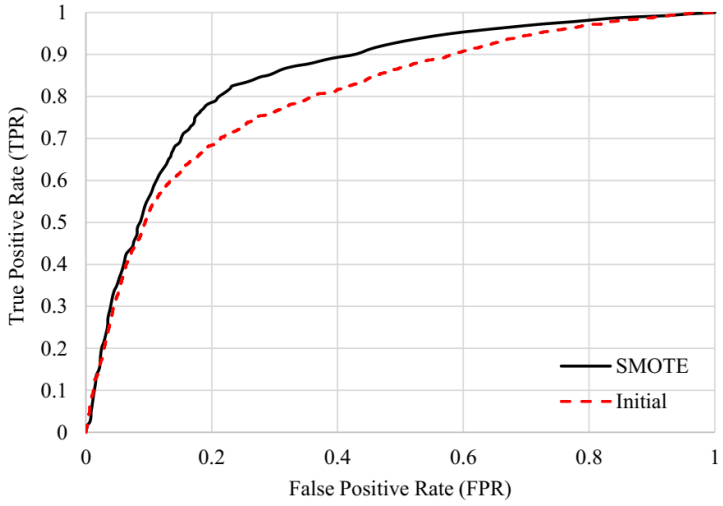


Figure 7.4: Comparison between the results obtained before and after incorporating SMOTE

Fig. 7.4 compares the ROC (and the corresponding AUC) of the Random Forest Failure detection model before and after applying SMOTE. Here, the area under curve (AUC) for the results obtained via SMOTE is significantly larger than the AUC obtained without SMOTE.

In order to ascertain an appropriate prediction method for this analysis, three additional models (SVM, NN Classifer and Decision Tree) were also implemented as part of the ED Meta-Learner. A *Support Vector Machine* (SVM) is a discriminative classification technique that incorporates a hyper-plane to distinguish between classes. SVM works based on mapping the data into a higher dimensional data space in order to be able to create an optimal separating hyperplane in this space. The *Neural Network* (NN) classifier was also applied and is a member of neural network family.

The *Decision Tree* (DT) is a classification technique that incorporates a hierarchical tree structured decision making process to identify the correct class for a given sample. In this tree structure, each internal node processes one particular data attribute and based on the observed value, determines the next processing node with leaves representing the class labels.

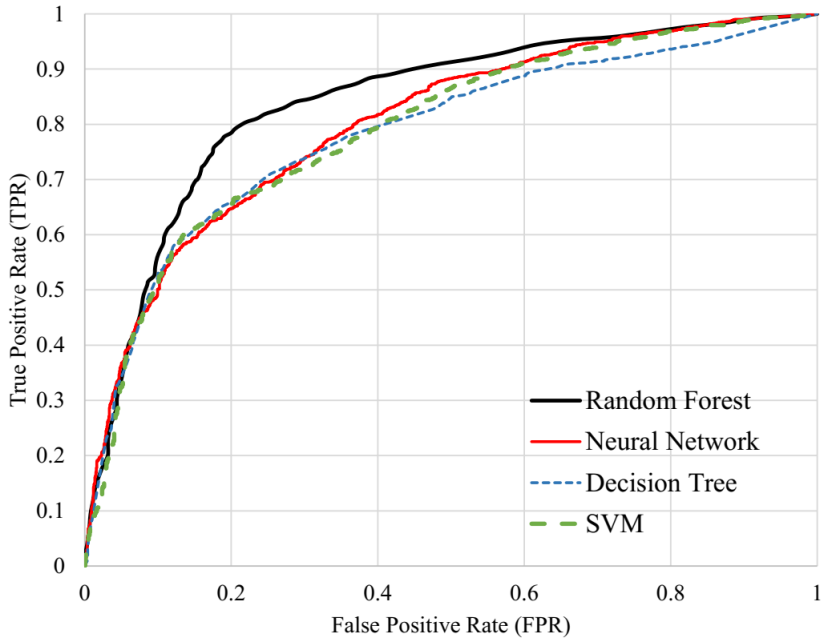


Figure 7.5: Compare ROC of all methods

For this part of our validation, SVM, NN and Decision tree classifiers together with SMOTE strategy for over sampling the data were also implemented, and the results of their ROC's are compared with the Random Forest early detection model in Fig. 7.5. As can be seen, the RF early detection model has outperformed the SVM, DT and NN classifiers and indicates a larger AUC in comparison with those models.

Fig. 7.6 also compares the confidence intervals of the ROCs of all the methods

Also Table 7.3 compares the significance of the AUC for all the methods in terms of min and max values.

The results shown in Fig. 7.6 and Table 7.3 were obtained in a bootstrapping process where experiments were repeated over five different train and test sets.

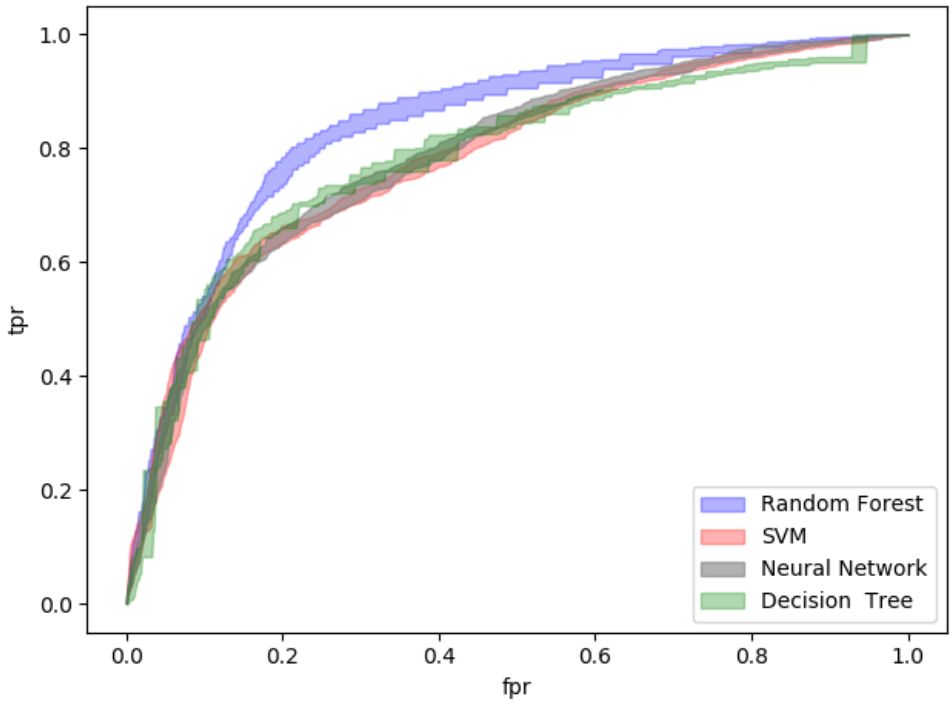


Figure 7.6: Compare ROC confidence intervals of all methods

Table 7.4 presents the results for RF, SV, NN and DT classifiers as used in implementations of the early detection model in terms of their confusion matrix elements. The Random Forest classifier has shown to have a significantly better prediction accuracy for the positive class in comparison with all other three methods, and its prediction accuracy for the negative class is 69.5% was comparable to Decision Tree 71.14% which showed the best accuracy.

In addition to the accuracy measure, the performance of a classifier can also be measured by calculating the area under the ROC curve or AUC. AUC corresponds to the probability that a random positive sample has a higher ranking than a random negative sample, resembling the two sample Wilcoxon rank-sum statistic [119].

Fig. 7.7 compares the accuracy and AUC of the classifiers used to implement our failure detection model. This figure shows that the RF classifier outperforms SVM, DT and NN classifiers in terms of *both* accuracy and AUC.

Table 7.3: Comparison between RF, SV, NN and DT classifiers in terms of the confusion matrix

Method	AUC min	AUC max
Random Forest	0.826	0.852
SVM	0.770	0.796
Decision Tree	0.771	0.798
Neural Network	0.777	0.800

Table 7.4: Comparison between RF, SV, NN and DT classifiers in terms of the confusion matrix

Method	TP	FP	TN	FN	Positive Accuracy	Negative Accuracy
Random Forest	927	172	1347	592	**84**%	69.5%
SVM	937	238	1281	582	79.74%	68.76%
Decision Tree	1047	355	1164	472	74.62%	**71.14**%
Neural Network	1031	356	1163	488	74.33%	70.44%

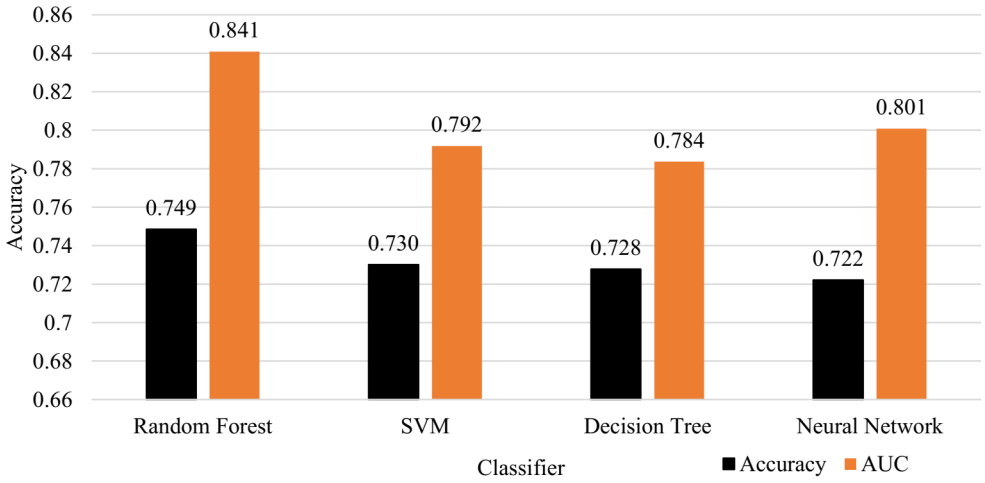


Figure 7.7: Comparison of the Accuracy and AUC of the each Classifier

7.2 MRFA Validation using a Large Time Series Dataset

In Chapter 4, we discovered that MRFA had a positive predictive ability when combined with Recurrent Neural Networks and then compared with a number of

popular time series prediction techniques such as Support Vector Regression and Neural Networks. However, this analysis was conducted on a small dataset. This work examined the effectiveness of MRFA when applied to a variety of machine learning techniques, but only over *20* time series. Since that study, we developed a methdology for creating a large, diverse synthetic time series dataset and are now in a position to review those findings using a more robust validation threshold. Effectively, this final study will combine the research developments from Chapters 4, 5 and 6 to develop a new validation process for multi-step ahead predictve algorithms. To be precise, we use the large dataset that was created for the meta-learning using the model/strategy combinations presented in Chapter 6.

We use a repeated measures model to examine the impact of strategy and method over time. The following methods and strategies were included in the analysis: REC^{ARIMA}, REC^{SVR}, REC^{NN}, REC^{RNN}, REC^{LSTM}, $MRFA^{SVR}$, $MRFA^{NN}$, and $MRFA^{RNN}$.

7.2.1 Overall Performance

A boxplot representation is used to inspect the overall performance variations of the candidate models. This offers a standardized method for representing the distribution of data with respect to a five factor summary of the data including: 1- Minimum, 2- First quartile (Q1), 3- Median, 4-Third quartile (Q3), and 5- Maximum. The box plot representation of the $\log nMSE$ prediction performance of the candidate models is shown in Fig. 7.8 for one step ahead prediction, in Fig. 7.9 for five steps ahead, Fig. 7.10 for 10 steps ahead, and Fig. 7.11 for 20 steps ahead.

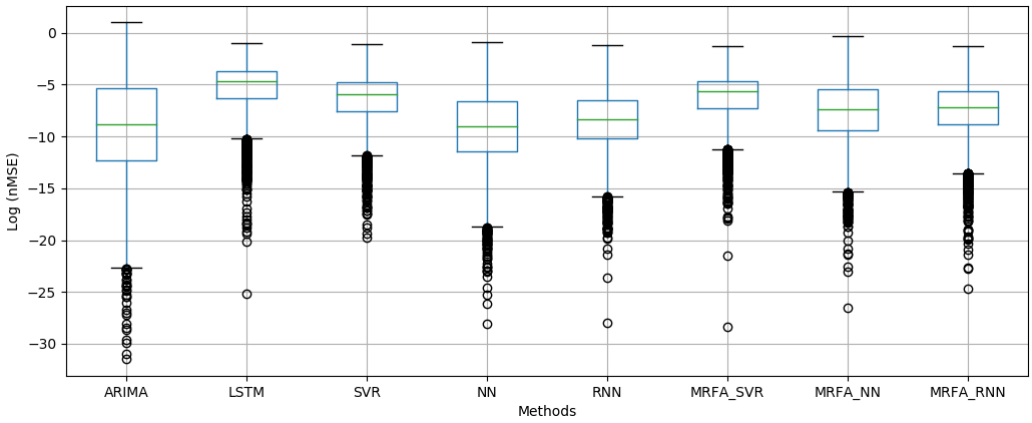


Figure 7.8: Box plot comparison for One Step Ahead Prediction

Fig. 7.8 shows that LSTM and NN exhibit the lowest and highest accuracy for one step ahead prediction, respectively. However, the distance between Q3 and Q1 was the lowest for LSTM indicating that LSTM has produced a smaller range of errors in comparison with others, for one step ahead prediction. $MRFA^{SVR}$ did not perform well in terms of the number of the series for which it has been identified as the best performing method. However, it showed a narrow *between-quantile* distance indicating that this model produced a *narrower* range of errors. In comparison with REC^{SVR}, $MRFA^{SVR}$ has a wider between-quantiles distance and also was less effective in terms of the number the series for which it was identified as the best performing method. $MRFA^{NN}$ showed a wider between-quantile distance and also a higher average $\log nMSE$ in comparison with REC^{NN} which means that $MRFA$ was not so effective in improving the performance for one step ahead prediction. $MRFA^{RNN}$ showed a slightly wider between-quantile distance and also a higher average $\log nMSE$ in comparison with REC^{RNN}, indicating that MRFA was not so successful in improving REC for one step ahead prediction.

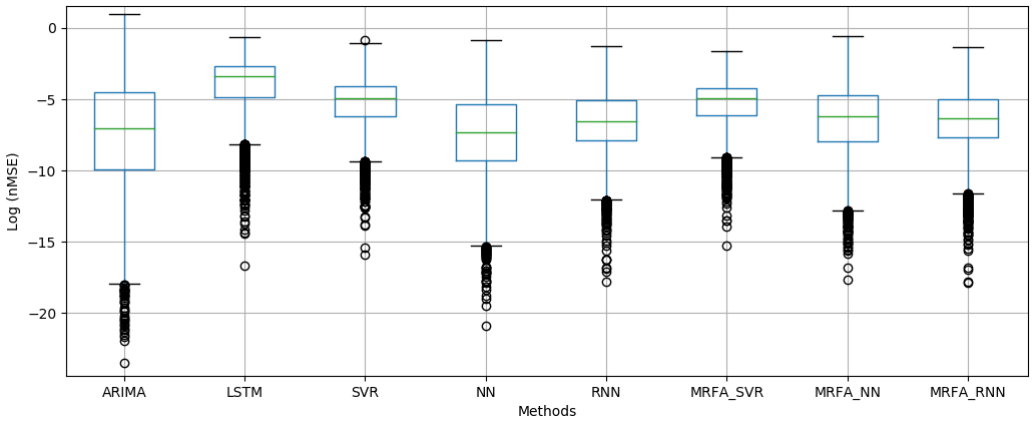


Figure 7.9: Box plot comparison for Five Steps Ahead Prediction

In Fig. 7.9, it can be seen that once again LSTM (REC^{LSTM}) has shown the worst performance, this time for 5 steps ahead prediction. This is as a result of the lack of sufficient training sample as LSTM (REC^{LSTM}) requires a large number of training samples to be trained properly. ARIMA (REC^{ARIMA}) shows a good performance in terms of the number of best performing occasions. However, the distance between its Q3 and Q1 was the longest among the candidate models for 5 steps ahead prediction and this indicates that ARIMA (REC^{ARIMA}) might show very bad performances also.

$MRFA^{NN}$ showed a narrower between-quantile distance but a higher average $\log nMSE$ in comparison with REC^{NN} which means that $MRFA$ was not so effective in improving the average performance for five steps ahead prediction, but performed slightly better in terms of the range of prediction errors produced. $MRFA^{RNN}$ showed a slightly narrower between-quantile distance, indicating that MRFA has shown a slightly better performance in comparison with REC in terms of the range of the produced prediction errors for one step ahead prediction.

$MRFA^{SVR}$ did not perform outperform REC^{SVR} in terms of the number of the series for which it has been identified as the best performing method and showed the same result. However, it showed a narrower between-quantiles distance indicating that this model has produced a narrower range of prediction errors REC^{SVR}. In

addition, $MRFA^{SVR}$ has shown the shortest distance between its Q3 and Q1 which means that $MRFA^{SVR}$ has exhibited the most *stable* performance for five steps ahead prediction.

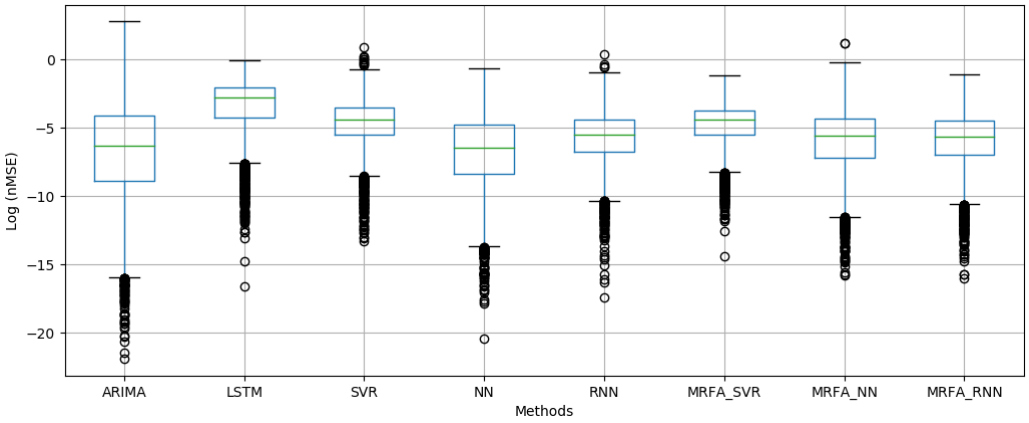


Figure 7.10: Box plot comparison for 10 steps ahead prediction

Fig. 7.10 shows that, again, REC^{ARIMA} exhibits the widest distance between Q3 and Q1 while have been the best performing approach for quite a bit number of series. REC^{SVR}, REC^{RNN} and $MRFA^{RNN}$ represented the narrowest between-quantiles distances (the distance between Q3 and Q1) for 10 steps ahead prediction. $MRFA^{SVR}$ and $MRFA^{RNN}$ have shown improved between-quantiles distance and the average $\log nMSE$ in comparison with REC^{SVR} and REC^{RNN}, respectively for 10 steps ahead prediction. However, $MRFA^{NN}$ did not improve REC^{NN}, not in terms of the average $\log nMSE$ nor the between-quantile distance.

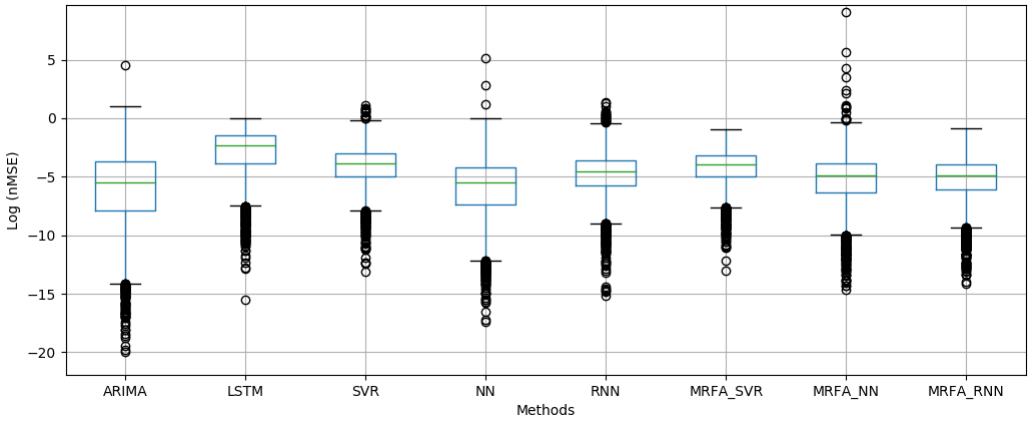


Figure 7.11: Box plot comparison for 20 steps ahead prediction

Finally, in Fig. 7.11, the Boxplot figure shows the same results for REC^{ARIMA} this time for 20 steps ahead prediction, i.e., being the best performing method for a large number of series but with the widest between-quantiles distance. $MRFA^{NN}$ did not improve REC^{NN}, not in terms of the average $\log nMSE$ nor the between-quantile distance. However, $MRFA^{SVR}$ and $MRFA^{RNN}$ have shown improved between-quantiles distance and the average $\log nMSE$ in comparison with REC^{SVR} and REC^{RNN}, respectively for 20 steps ahead prediction.

To better compare the prediction methods on the created time series, we used a heatmap representing the min, mean and max values for each categories presented in Chapter 5, shown in Fig. 7.12.

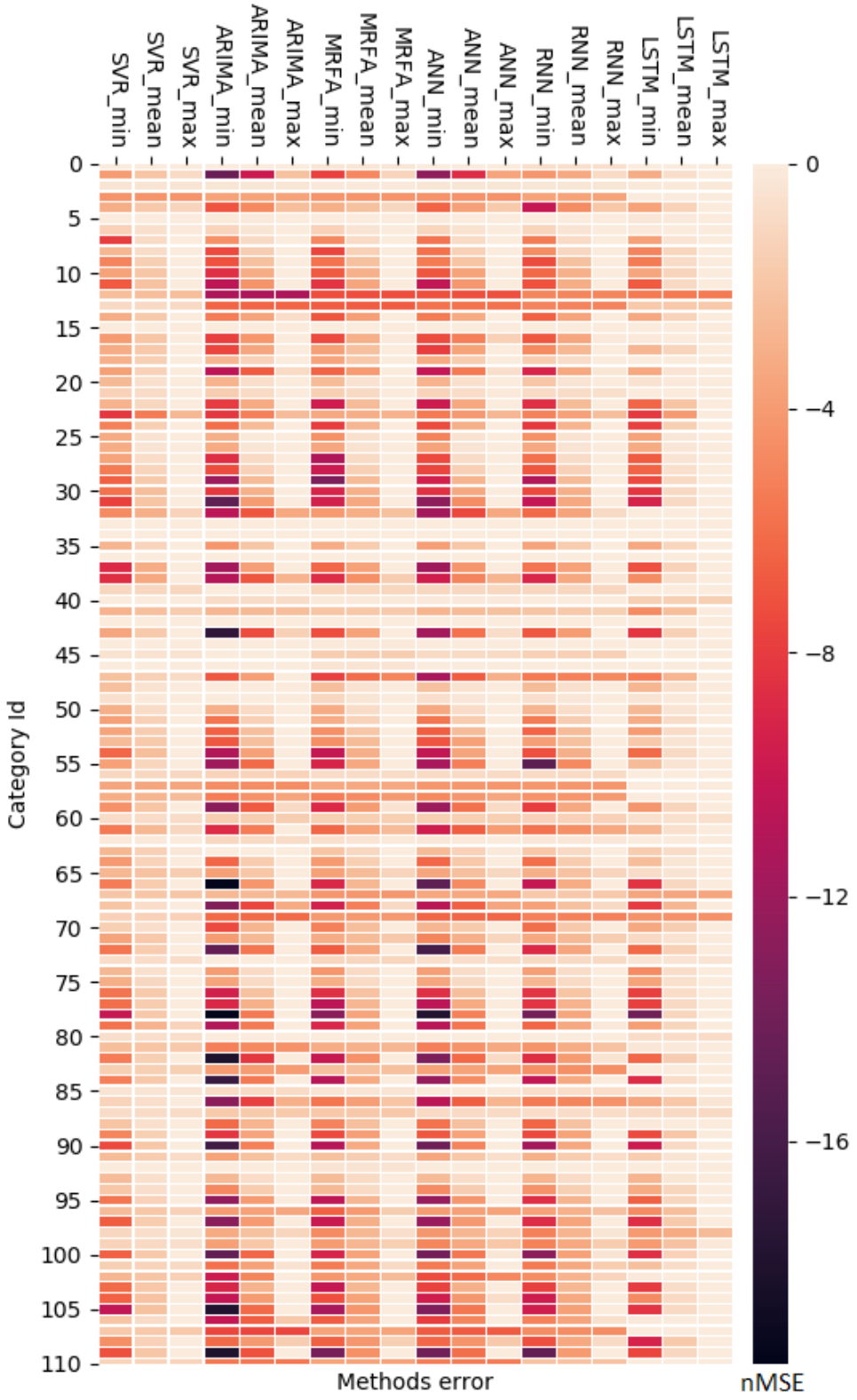


Figure 7.12: Compare the methods for mean, min and max for different categories of time series

7.2.2 Analyzing the Prediction Horizon

While the boxplots in the previous section allowed us to inspect the overall performance, this analysis did not measure the capabilities of each model over the *entire prediction horizon*. It is important to examine the changes to the $\log nMSE$ value for each prediction model for the entire prediction horizon to understand how the performances of the prediction models change *over time*. This will also help us to compare the prediction models in a continuous fashion and have a visual understanding of the impact of time on the performance of each prediction model.

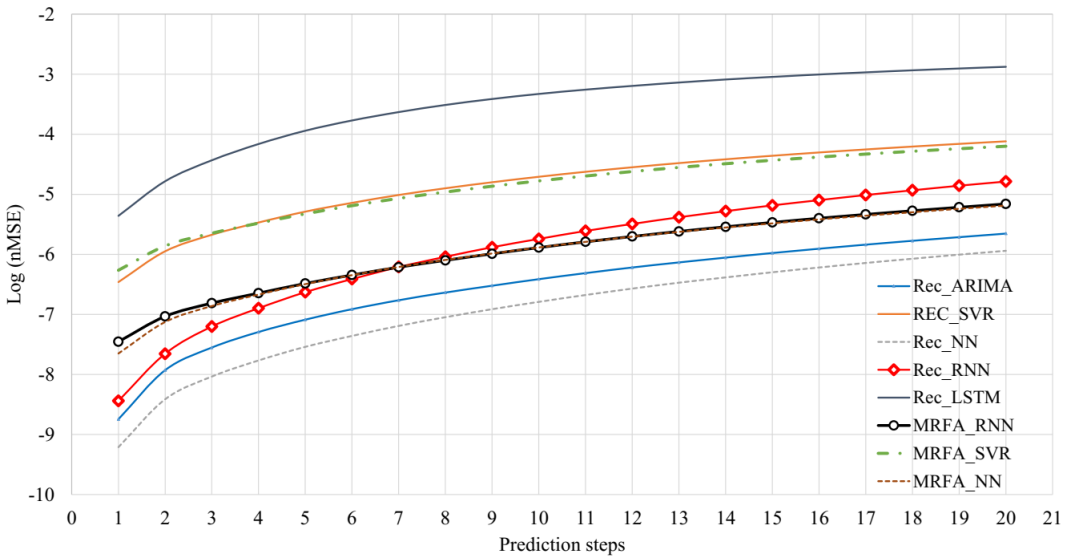


Figure 7.13: Comparison between REC and $MRFA$ for all possible models

Fig. 7.13 compares all candidate models introduced in Table 6.2 in terms of the mean $\log nMSE$ for the entire prediction horizon. The uncertainty bands for each curve in Fig. 7.13 have been depicted in Fig. 7.14 to Fig. 7.21.

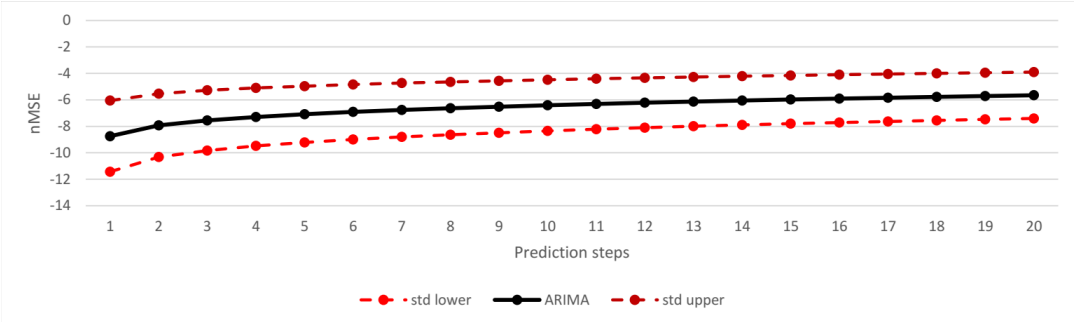


Figure 7.14: ARIMA with uncertainty bands

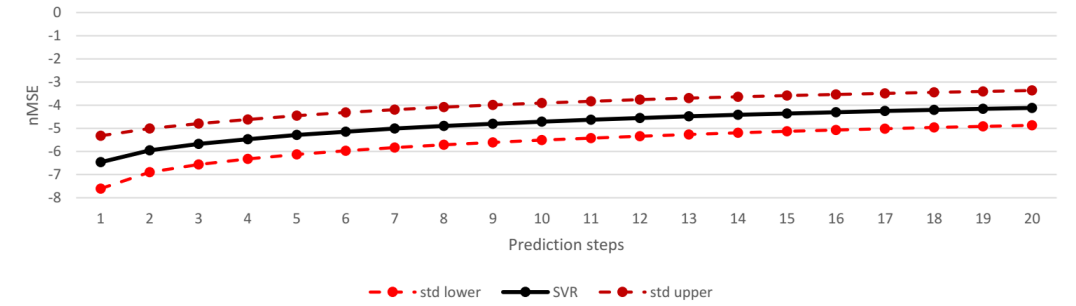


Figure 7.15: ARIMA with uncertainty bands

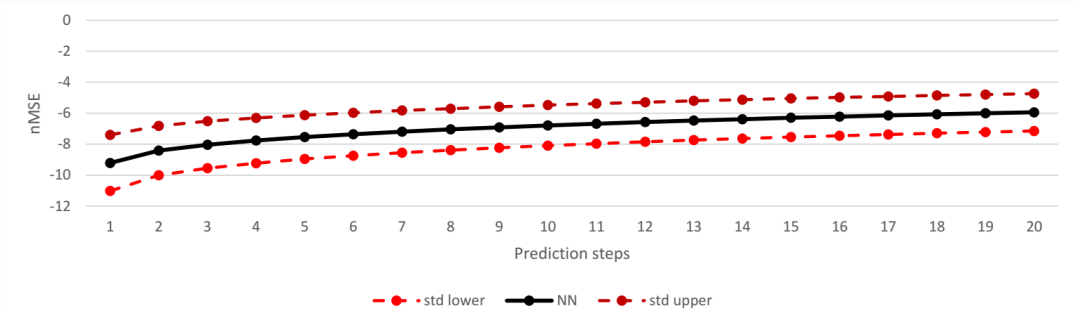


Figure 7.16: NN with uncertainty bands

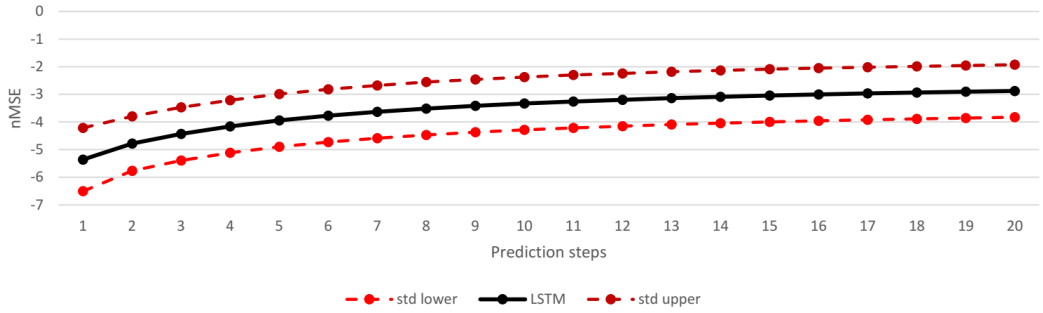


Figure 7.17: LSTM with uncertainty bands

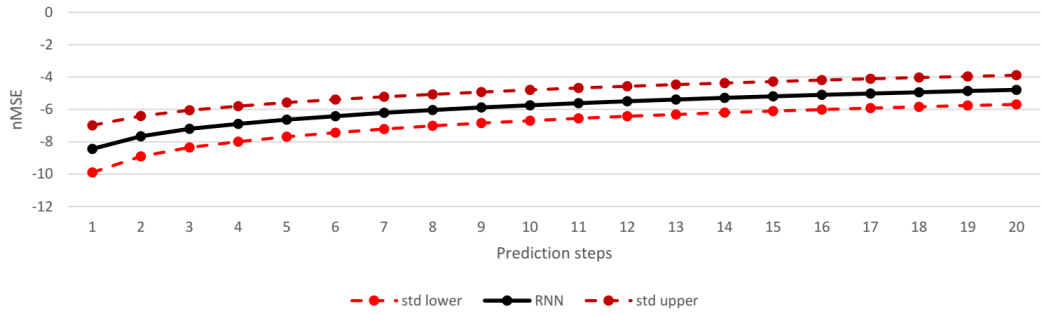


Figure 7.18: RNN with uncertainty bands

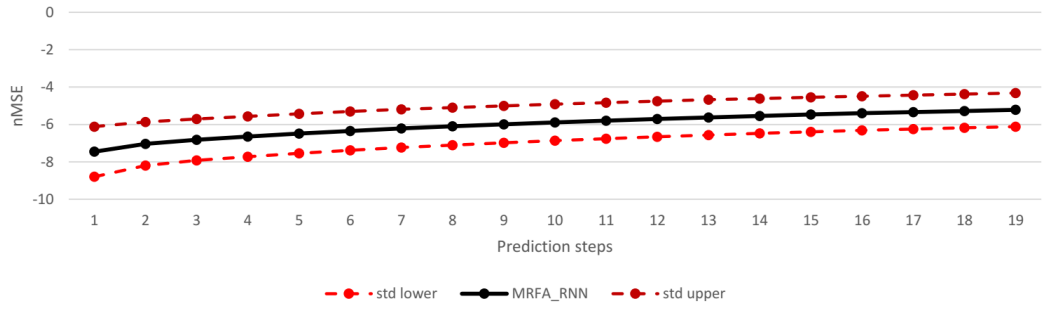


Figure 7.19: $MRFA^{RNN}$ with uncertainty bands

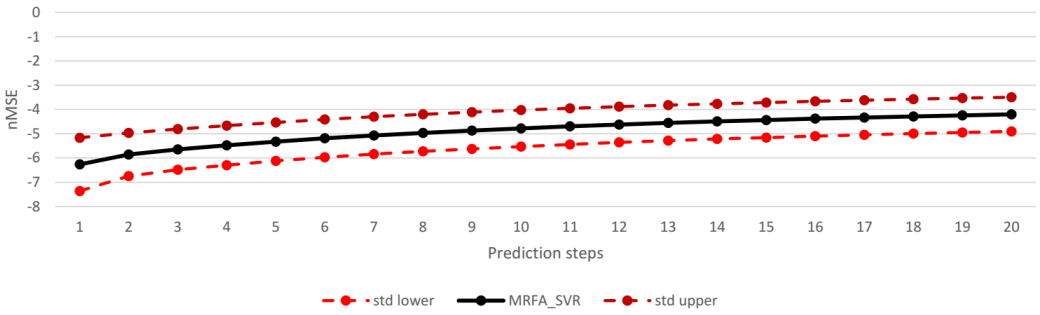


Figure 7.20: $MRFA^{SVR}$ with uncertainty bands

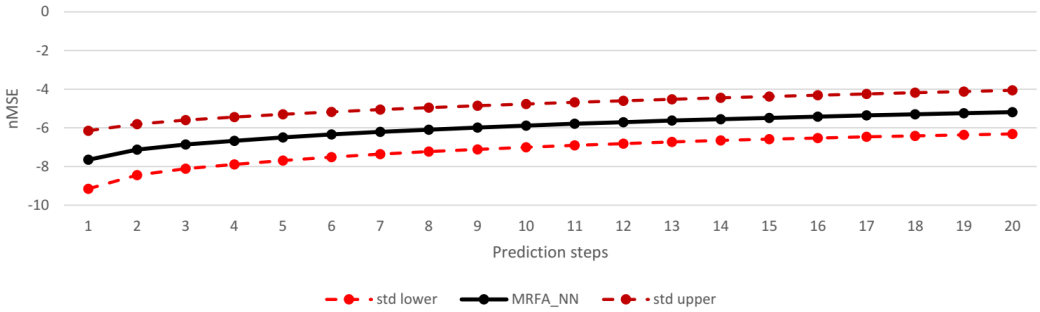


Figure 7.21: $MRFA^{NN}$ with uncertainty bands

A comparison between the recursive strategy (REC) and MRFA in terms of mean $\log nMSE$ when implemented using RNN for the entire prediction horizon is provided in Fig. 7.22.

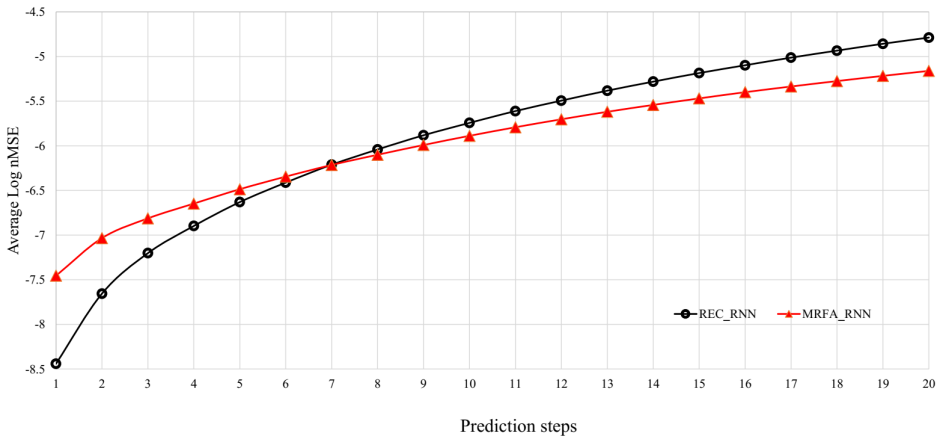


Figure 7.22: Comparison between REC^{RNN} and $MRFA^{RNN}$

Fig. 7.22 shows that $MRFA^{RNN}$ improves REC^{RNN} after 7 prediction steps. However, in order to understand if there was a significant improvement over time for any of the methods/strategies, a repeated measures model was implemented to examine if there was an interaction between time and the method/strategy combination. For this analysis, both strategy and method were treated as the *between subject* effects and the prediction step was treated as the repeated measure or *within subject* effect. Time series features are also needed to be included as covariates in the analysis.

We used Principal Component Analysis to reduce the complexity and volume of data; PCA is an approach to reduce the dimensionality of the data. In this experiment, three principal component variables were created from the original continuous time series features and treated as covariates in the analysis. The results of the analysis showed that there was a significant interaction between time, strategy and model using a Greenhouse-Geisser test on the within subject effects [98], i.e., $p < 0.001, F_{6.346,98288} = 74.517$. The Greenhouse-Geisser test is an approach for measuring the lack of sphericity in a repeated measures ANOVA. A detailed analysis of all the significant effects is provided in Appendix A. The Greenhouse-Geisser test was used as the assumption of sphericity was violated $Mauchly's W_{189} = 0., p < 0.001$. This shows that there is a significant interaction between time, model and strategy which indicates that the choice of strategy is important in the performance of the prediction model for the given time series.

Fig. 7.23 shows the estimated Marginal Mean Difference between MRFA and REC by Prediction Step.

Fig. 7.23, for each candidate model (NN, RNN and SVR), a separate line is considered representing the difference between the average nMSE obtained by MRFA and the recursive strategy (REC); or $MRFA^{model} - REC^{model}$ where model can be NN, RNN or SVR.

In Fig. 7.23, we see that the estimated nMSE marginal mean difference between MRFA and REC reduces as the prediction horizon extends. REC outperforms MRFA on NN over the 20-step prediction horizon. However, MRFA can be seen

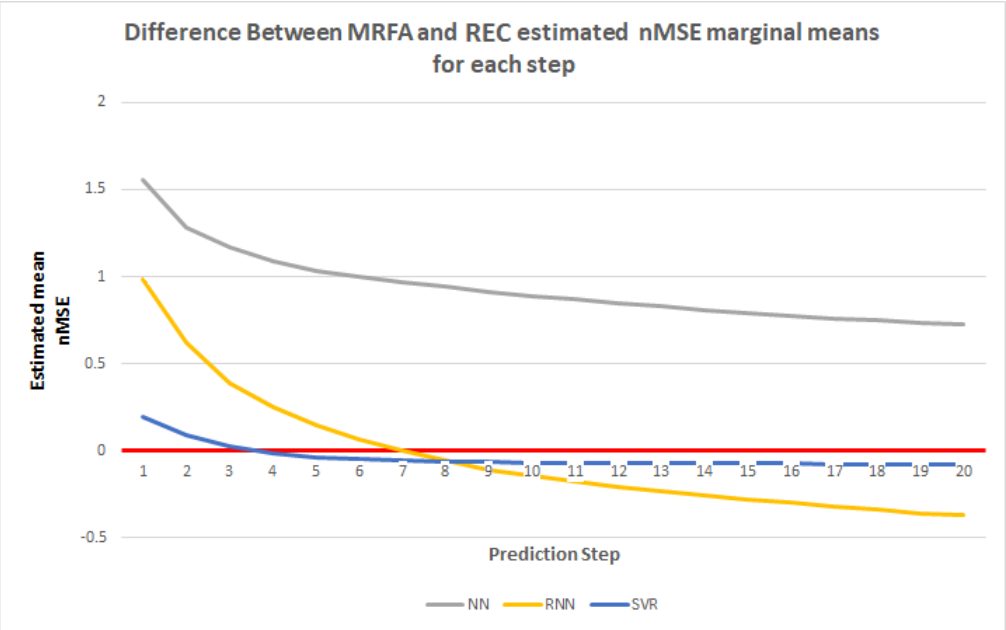


Figure 7.23: Estimated Marginal Mean Difference between MRFA and REC by Prediction Step

to perform well as the prediction horizon extends further in time for both RNN and SVR models. As indicated by the yellow line, for RNN, MRFA shows a better performance than REC after 8 steps since $MRFA^{RNN} - REC^{RNN} < 0$ after 8 steps

7.3 Summary

The goal of this Chapter was to combine the three research developments presented in this dissertation to carry out a number of case studies to further present the impact of the research. We first proposed an early detection meta-learner that identifies potentially weak candidate prediction models for the given time series. Our results showed that this (failure seeking) meta-learner was very successful in detecting weak or inappropriate predictive models.

Our second study was concerned with a deeper evaluation of our MRFA model and in particular, analyzing the entire prediction horizon. A number of analyses were incorporated to examine if there was an interaction between time and the

method/strategy combination when applied to the time series introduced in Chapter 5. In particular, a repeated measures model was implemented to help us to understand if there was a significant improvement over time for any of the methods/strategies. The results of the Greenhouse-Geisser test showed that REC outperforms MRFA on NN over the 20-step prediction horizon. However, MRFA out performs REC as the prediction horizon extends further in time for both RNN and SVR.

Chapter 8

Conclusions

This final Chapter has 2 parts: in section 8.1, a summary of the work completed for this dissertation is presented and in section 8.2, we discuss some areas for future research projects that could extend this area of research.

8.1 Dissertation Overview

This dissertation is focused on predictive models for time series data, the difficulty in selecting appropriate models for different types of time series data, and how to ensure robust evaluations for new predictive models. In §1.1, we provided an introduction to Time Series predictive algorithms before discussing some open research issues which helped to articulate our hypothesis in which three research questions were posed. This first research question asks how the integration of metrics derived from multi-resolutional sliding windows together with a recursive MSAP strategy can be used to reduce error accumulation. The second question asked if a large number of synthetic time series could be developed with a sufficiently diverse set of time series characteristics. The final question asked if it was possible to build a meta-learner which could help time series researchers select the predictive model best suited to the dataset under review.

The literature review in Chapter 2 covered Multi-step Ahead Prediction, synthetic time series generation and meta-learning. While several MSAP strategies were discussed, the accumulation of error in REC based strategies and the intermediate information loss in DIR mean that this is still an open area for researchers. We also learnt that the existing synthetic time series generation approaches failed to provide datasets sufficiently diverse to assess time series models. When reviewing meta-learning research, it revealed that existing approaches fail to address hyper-parameter selection in machine learning models.

We then presented our recursive approach, Multiple Resolution Forecast Aggregation, that addressed the shortcomings of the original REC model by using the *Resolutions of Impact* concept. Our validation used 20 time series and provided a detailed study on the behaviour of Irish Pig Price data. This evaluation showed that the preferred method for certain models depends on the *characteristics* of the underlying time series. This led to the conclusion that a large number of time series, comprising a more diverse range of characteristics, is necessary when developing new predictive models. This highlighted a requirement to construct a large numbers of time series with the appropriately diverse range of characteristics.

The approach to synthetic data construction comprised 5 well-known time series features and used a multivariate entropy measure to confirm the diversity of the created time series. The experimental results showed that our overall dataset measured diversity at 83.4%, which delivered a solid contribution. As part of this step, we developed the *coverage rate* metric to measure the coverage of the dataset over the full feature space. The results for this part of the evaluation delivered a coverage rate of 72%, which we felt was another significant contribution given the size of the time series dataset.

For the final contribution of this dissertation, we presented a meta-learning approach for selecting the appropriate method for MSAP time series prediction using a set of time series features. Unlike existing approaches that use classifiers, we incorporated a regression method that estimated the log-based normalized Mean Squared Error, $\log nMSE$. To train our meta-learner, we ensured that the dataset was sufficiently

diverse and of a large enough size to provide a solid generalization ability for our meta-learning approach. The R of the resulted predictions indicate that our meta-learner has 94% accuracy which should provide significant benefit to time series researchers who are trying to narrow the field in terms of model selection.

In the final Chapter, we carried out two case studies. First we developed a early prediction meta-learner using the Ensemble Random Forest classifier technique to identify weak or inappropriate predictive models, with very positive results. This can have a strong impact for time series researchers in that it could save a lot of time running evaluations for models that will never deliver good results. Secondly, we analyzed the results to investigate if our MRFA strategy improves REC with RNN in long prediction horizons. A number of analyses were incorporated to examine if there was an interaction between time and the method/strategy combination. Once again, there were positives to be taken from the results as our validation showed MRFA performs well as the prediction horizon extends further in time for both RNN and SVR. This provides researchers with a clear experimental plan when using either MRFA or recurrent neural networks.

8.2 Future Research

In this section, we provide some ideas for extending the research presented in this dissertation.

Our goal when proposing MRFA was to improve the performance of Multi-step ahead prediction. However, during the implementation of the algorithm and also during the experiments, we identified areas of research where further improvements could be made.

MRFA could be improved by developing a mechanism that can adapt MRFA's parameters such as the contribution weights of the RNNs, with the temporal changes of the statistical properties of the time series.

LSTM and its variants, such the Gated Recurrent Unit (GRU), are an advanced

version of RNN that provide enhanced abilities in capturing nonlinear structures in the time series, such as long term memory. Implementing LSTM with MRFA could enhance the capacity of MRFA in dealing with non-linearity and complexity in time series prediction problems.

We can use the variance of the output signal as a useful measure the data uncertainty and incorporate it to improve the performance of MRFA.

Despite having achieved a relatively high level of diversity, future work should consider the inclusion of additional features such as *chaos*. Additionally, in Chapter 5 we demonstrated that the coverage of the feature space was 72%, however, there was a wide range in the density values of the feature space categories. The meta-learner would yield improved predictive power if we narrowed the density value range between all the applicable categories.

In Chapter 6, we introduced a methodology for identifying the appropriate method for a given time series. However, there still exists a number of limitations, and thus opportunities for further improvements in future research.

The meta-learning in this research was trained on a relatively small dataset (5,819 time series), with a coverage rate of 30% and a diversity (multivariate entropy) of 59%. In order to fully understand the predictive power of each algorithm a dataset should be selected using a sampling strategy which would return a much higher level of diversity.

In addition, we only used 5 prediction **models** within our $Strategy^{model}$ combinations, and these included ARIMA, NN, RNN, SVR and LSTM. The versions used in this analysis were the vanilla versions from the respective families. For example, we used the vanilla RNN which is the prototype of RNN models and its advanced versions offer more dynamic structures that can improve prediction performance. However in some cases, these advanced versions have a limited use as they can potentially cause over-fitting when applied to relatively small time series datasets.

Although we offered a solution on how to manage hyper-parameters, hyper-parameter optimization is still a major challenge in the machine learning community, and we cannot be sure that they were chosen optimally in our experiments. As mentioned in Chapter 6, hyper-parameter optimization in machine learning techniques uses a grid search algorithm when searching for an optimum solution. This is a very time consuming and computationally expensive and would benefit greatly from the availability of extensive high performance computing facilities

Bibliography

[1] *kaggle*. http://www.kaggle.com.

[2] *UCI*. http://archive.ics.uci.edu.

[3] Ratnadip Adhikari and RK Agrawal. A combination of artificial neural network and random walk models for financial time series forecasting. *Neural Computing and Applications*, 24(6):1441–1449, 2014.

[4] Monica Adya, Fred Collopy, J Scott Armstrong, and Miles Kennedy. Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting*, 17(2):143–157, 2001.

[5] Ida Kalate Ahani, Majid Salari, and Alireza Shadman. Statistical models for multi-step-ahead forecasting of fine particulate matter in urban areas. *Atmospheric Pollution Research*, 10(3):689–700, 2019.

[6] Ida Kalate Ahani, Majid Salari, and Alireza Shadman. An ensemble multi-step-ahead forecasting system for fine particulate matter in urban areas. *Journal of Cleaner Production*, page 120983, 2020.

[7] Sabri Ahmad et al. Forecasting on the crude palm oil and kernel palm production: Seasonal arima approach. In *2011 IEEE Colloquium on Humanities, Science and Engineering*, pages 939–944. IEEE, 2011.

[8] Zainy MH Almurad and Didier Delignières. Evenly spacing in detrended fluctuation analysis. *Physica A: Statistical Mechanics and its Applications*, 451:63–69, 2016.

[9] Orly Alter, Patrick O Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.

[10] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 188–193. IEEE, 2017.

[11] Nguyen Hoang An and Duong Tuan Anh. Comparison of strategies for multi-step-ahead prediction of time series using neural network. In *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pages 142–149. IEEE, 2015.

[12] Nikkie C Anand, Caterina Scoglio, and Balasubramaniam Natarajan. Garch—non-linear time series model for traffic modeling and prediction. In *NOMS 2008-2008 IEEE Network Operations and Management Symposium*, pages 694–697. IEEE, 2008.

[13] Nicholas Apergis and Anthony Rezitis. Food price volatility and macroeconomic factors: Evidence from garch and garch-x estimates. *Journal of Agricultural and Applied Economics*, 43(1):95–110, 2011.

[14] B Arinze. Selecting appropriate forecasting models using rule induction. *Omega*, 22(6):647–658, 1994.

[15] J Scott Armstrong. Selecting forecasting methods. In *Principles of forecasting*, pages 365–386. Springer, 2001.

[16] Mariette Awad and Rahul Khanna. Support vector regression. In *Efficient Learning Machines*, pages 67–80. Springer, 2015.

[17] James Aweya. *Sensitivity Methods for Congestion Control in Computer Networks*. PhD thesis, School of computing, Ottawa, Ont., Canada, Canada, 1999. AAINQ48085.

[18] Honey Badrzadeh, Ranjan Sarukkalige, and AW Jayawardena. Impact of multi-resolution analysis of artificial intelligence models inputs on multi-step ahead river flow forecasting. *Journal of Hydrology*, 507:75–85, 2013.

[19] Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines. Simulated data experiments for time series classification part 1: accuracy comparison with default settings. *arXiv preprint arXiv:1703.09480*, 2017.

[20] Fouad Bahrpeyma. *Diversity in 50K Time Series*, 2020. https://drive.google.com/drive/folders/12DfKOkdePtltCN5761BKX6oYRwGc83GM?usp=sharing.

[21] Fouad Bahrpeyma, Mark Roantree, and Andrew McCarren. Multi-resolution forecast aggregation for time series in agri datasets. In *Proceedings of the 25th Irish Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland, December 7 - 8, 2017*, volume 2086 of *CEUR Workshop Proceedings*, pages 193–205. CEUR-WS.org, 2017.

[22] Fouad Bahrpeyma, Mark Roantree, and Andrew McCarren. Multistep-ahead prediction: A comparison of analytical and algorithmic approaches. In *DaWaK 2018 : 20th International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2018.

[23] Fouad Bahrpeyma, Ali Zakerolhoseini, and Hassan Haghighi. Using ids fitted q to develop a real-time adaptive controller for dynamic resource provisioning in cloud's virtualized environment. *Applied Soft Computing*, 26:285–298, 2015.

[24] Tea Baldigara and Maja Mamula. Modelling international tourism demand using seasonal arima models. *Tourism and hospitality management*, 21(1):19–31, 2015.

[25] Cristiano Ballabio. Spatial prediction of soil properties in temperate mountain regions using support vector regression. *Geoderma*, 151(3-4):338–350, 2009.

[26] Yidan Bao, Haiyan Cen, Yong He, and Lilan Lin. Application of improved bp neural network to predict agricultural commodity total production value. In *2006 International Conference on Computational Intelligence and Security*, volume 2, pages 992–995. IEEE, 2006.

[27] Yukun Bao, Tao Xiong, and Zhongyi Hu. Pso-mismo modeling strategy for multistep-ahead time series prediction. *IEEE transactions on cybernetics*, 44(5):655–668, 2013.

[28] Yukun Bao, Tao Xiong, and Zhongyi Hu. Multi-step-ahead time series prediction using multiple-output support vector regression. *Neurocomputing*, 129:482–493, 2014.

[29] Sasan Barak, Mahdi Nasiri, and Mehrdad Rostamzadeh. Time series model selection with a meta-learning approach; evidence from a pool of forecasting algorithms. *arXiv preprint arXiv:1908.08489*, 2019.

[30] TG Barbounis and John B Theocharis. Locally recurrent neural networks for wind speed prediction using spatial correlation. *Information Sciences*, 177(24):5775–5797, 2007.

[31] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[32] E Benito, A Stiggelbout, FX Bosch, A Obrador, J Kaldor, M Mulet, and N Munoz. Nutritional factors in colorectal cancer risk: a case-control study in majorca. *International Journal of Cancer*, 49(2):161–167, 1991.

[33] Jan Beran. *Statistics for long-memory processes*. Routledge, 2017.

[34] Parag P Bhagwat, Rajib Maity, et al. Multistep-ahead river flow prediction using ls-svr at daily scale. *Journal of water Resource and Protection*, 4(07):528, 2012.

[35] Vince J Bidwell. Realistic forecasting of groundwater level, based on the eigenstructure of aquifer dynamics. *Mathematics and Computers in Simulation*, 69(1):12–20, 2005.

[36] Vince J Bidwell. Realistic forecasting of groundwater level, based on the eigenstructure of aquifer dynamics. *Mathematics and Computers in Simulation*, 69(1-2):12–20, 2005.

[37] Baki Billah, Maxwell L King, Ralph D Snyder, and Anne B Koehler. Exponential smoothing model selection for forecasting. *International journal of forecasting*, 22(2):239–247, 2006.

[38] Mauro Birattari, Gianluca Bontempi, and Hugues Bersini. Lazy learning meets the recursive least squares algorithm. In *Advances in neural information processing systems*, pages 375–381, 1999.

[39] Søren Bisgaard and Murat Kulahci. *Time series analysis and forecasting by example*. John Wiley & Sons, 2011.

[40] Adriaan M Bloem, Robert Dippelsman, Nils O Maehle, and Nils Øyvind Mæhle. *Quarterly national accounts manual: concepts, data sources, and compilation*. International Monetary Fund, 2001.

[41] Peter Bloomfield. *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

[42] Gianluca Bontempi. Long term time series prediction with multi-input multi-output local learning. *Proc. 2nd ESTSP*, pages 145–154, 2008.

[43] Gianluca Bontempi. Long term time series prediction with multi-input multi-output local learning. 2008.

[44] Gianluca Bontempi and Souhaib Ben Taieb. Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International journal of forecasting*, 27(3):689–699, 2011.

[45] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. *Machine learning strategies for time series forecasting*, pages 62–77. Springer, 2013.

[46] George EP Box and Gwilym M Jenkins. *Time series analysis: forecasting and control, revised ed.* Holden-Day, 1976.

[47] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control.* John Wiley and Sons, 2015.

[48] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control.* John Wiley and Sons, 2015.

[49] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

[50] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[51] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting.* springer, 2016.

[52] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting.* springer, 2016.

[53] Morton B Brown and Alan B Forsythe. Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346):364–367, 1974.

[54] Heriberto Cabezas and Brian D Fath. Towards a theory of sustainable systems. *Fluid phase equilibria*, 194:3–14, 2002.

[55] M Carbon and M Delecroix. Non-parametric vs parametric forecasting in time series: A computational point of view. *Applied stochastic models and data analysis*, 9(3):215–229, 1993.

[56] John C Chambers, Satinder K Mullick, and Donald D Smith. *How to choose the right forecasting technique.* Harvard University, Graduate School of Business Administration, 1971.

[57] Christopher Chatfield. *The analysis of time series: theory and practice*. Springer, 2013.

[58] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[59] Jie Chen, Guo-Qiang Zeng, Wuneng Zhou, Wei Du, and Kang-Di Lu. Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization. *Energy Conversion and Management*, 165:681–695, 2018.

[60] Mingang Chen and Pan Liu. Performance evaluation of recommender systems. *International Journal of Performability Engineering*, 13(8):1246–1256, 2017.

[61] Yanhui Chen, Chuan Zhang, Kaijian He, and Aibing Zheng. Multi-step-ahead crude oil price forecasting using a hybrid grey wave model. *Physica A: Statistical Mechanics and its Applications*, 501:98–110, 2018.

[62] Yize Chen, Yishen Wang, Daniel Kirschen, and Baosen Zhang. Model-free renewable scenario generation using generative adversarial networks. *IEEE Transactions on Power Systems*, 33(3):3265–3275, 2018.

[63] Chun-Tian Cheng, Jing-Xin Xie, Kwok-Wing Chau, and Mehdi Layeghifard. A new indirect multi-step-ahead prediction model for a long-term hydrologic prediction. *Journal of Hydrology*, 361(1-2):118–130, 2008.

[64] Haibin Cheng, Pang-Ning Tan, Jing Gao, and Jerry Scripps. Multistep-ahead time series prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774. Springer, 2006.

[65] Chao-Hsien Chu and Djohan Widjaja. Neural network system for forecasting method selection. *Decision Support Systems*, 12(1):13–24, 1994.

[66] Michael Clements and David Hendry. *Forecasting economic time series*. Cambridge University Press, 1998.

[67] Antonio J Conejo, Miguel A Plazas, Rosa Espinola, and Ana B Molina. Day-ahead electricity price forecasting using the wavelet transform and arima models. *IEEE transactions on power systems*, 20(2):1035–1042, 2005.

[68] Peter F Craigmile. Simulating a class of stationary gaussian processes using the davies–harte algorithm, with application to long memory processes. *Journal of Time Series Analysis*, 24(5):505–511, 2003.

[69] Ralph B d'Agostino. An omnibus test of normality for moderate and large size samples. *Biometrika*, 58(2):341–348, 1971.

[70] Jessamyn Dahmen and Diane Cook. Synsys: A synthetic data generation system for healthcare applications. *Sensors*, 19(5):1181, 2019.

[71] Ioannis N Daliakopoulos, Paulin Coulibaly, and Ioannis K Tsanis. Groundwater level forecasting using artificial neural networks. *Journal of hydrology*, 309(1-4):229–240, 2005.

[72] Ali F Darrat and Maosen Zhong. On testing the random-walk hypothesis: a model-comparison approach. *Financial Review*, 35(3):105–124, 2000.

[73] Dipankar Dasgupta. Artificial neural networks and artificial immune systems: similarities and differences. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 1, pages 873–878. IEEE, 1997.

[74] Robert B Davies and DS Harte. Tests for hurst effect. *Biometrika*, 74(1):95–101, 1987.

[75] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[76] Thomas Dietterich. Machine learning for sequential data: A review. *Structural, syntactic, and statistical pattern recognition*, pages 227–246, 2002.

[77] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[78] Thomas G Dietterich. Machine learning for sequential data: A review. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pages 15–30. Springer, 2002.

[79] Yagob Dinpashoh, Rasoul Mirabbasi, Deepak Jhajharia, Hamid Zare Abianeh, and Ali Mostafaeipour. Effect of short-term and long-term persistence on identification of temporal trends. *Journal of Hydrologic Engineering*, 19(3):617–625, 2014.

[80] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[81] Paul Doukhan, George Oppenheim, and Murad Taqqu. *Theory and applications of long-range dependence*. Springer Science & Business Media, 2002.

[82] Claude Duchon and Robert Hale. *Time series analysis in meteorology and climatology: an introduction*, volume 7. John Wiley & Sons, 2012.

[83] Grzegorz Dudek. Neural networks for pattern-based short-term load forecasting: A comparative study. *Neurocomputing*, 205:64–74, 2016.

[84] Andreas Efstratiadis, Yannis G Dialynas, Stefanos Kozanis, and Demetris Koutsoyiannis. A multivariate stochastic model for the generation of synthetic time series at multiple time scales reproducing long-term persistence. *Environmental Modelling & Software*, 62:139–152, 2014.

[85] Robert Engle. Garch 101: The use of arch/garch models in applied econometrics. *The Journal of Economic Perspectives*, 15(4):157–168, 2001.

[86] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[87] Jianqing Fan and Qiwei Yao. *Nonlinear time series: nonparametric and parametric methods*. Springer Science & Business Media, 2008.

[88] Willliam Feller. *An introduction to probability theory and its applications, vol 2*. John Wiley & Sons, 2008.

[89] Rigoberto Fonseca-Delgado and Pilar Gomez-Gil. Selecting and combining models with self-organizing maps for long-term forecasting of chaotic time series. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 2616–2623. IEEE, 2014.

[90] Philip Hans Franses, Dick Van Dijk, et al. *Non-linear time series models in empirical finance*. Cambridge University Press, 2000.

[91] Yakov Frayman, Bernard F Rolfe, and Geoffrey I Webb. Solving regression problems using competitive ensemble models. In *Australian Joint Conference on Artificial Intelligence*, pages 511–522. Springer, 2002.

[92] Everette S Gardner Jr. Exponential smoothing: The state of the art—part ii. *International journal of forecasting*, 22(4):637–666, 2006.

[93] David Gerbing. Time series components. *Portland State University*, page 9, 2016.

[94] Ciaran Gilbert, Jakob W Messner, Pierre Pinson, Pierre-Julien Trombe, Remco Verzijlbergh, Pim van Dorp, and Harmen Jonker. Statistical post-processing of turbulence-resolving weather forecasts for offshore wind power forecasting. *Wind Energy*, 23(4):884–897, 2020.

[95] Nina Golyandina and Anatoly Zhigljavsky. *Singular Spectrum Analysis for time series*. Springer Science & Business Media, 2013.

[96] Dilan Görür. *Nonparametric Bayesian discrete latent variable models for unsupervised learning*. PhD thesis, Berlin Institute of Technology, 2007.

[97] Alex Graves. Long short-term memory. In *Supervised sequence labelling with recurrent neural networks*, pages 37–45. Springer, 2012.

[98] Samuel W Greenhouse and Seymour Geisser. On methods in the analysis of profile data. *Psychometrika*, 24(2):95–112, 1959.

[99] Zhenhai Guo, Dezhong Chi, Jie Wu, and Wenyu Zhang. A new wind speed forecasting strategy based on the chaotic time series modelling technique and the apriori algorithm. *Energy Conversion and Management*, 84:140–151, 2014.

[100] Zhenhai Guo, Weigang Zhao, Haiyan Lu, and Jianzhou Wang. Multi-step forecasting for wind speed using a modified emd-based artificial neural network model. *Renewable Energy*, 37(1):241–249, 2012.

[101] Matej Gustin, Robert S McLeod, and Kevin J Lomas. Forecasting indoor temperatures during heatwaves using time series models. *Building and Environment*, 143:727–739, 2018.

[102] Coşkun Hamzaçebi, Diyar Akay, and Fevzi Kutay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications*, 36(2):3839–3844, 2009.

[103] Ping Han, Peng Xin Wang, Shu Yu Zhang, et al. Drought forecasting based on the remote sensing data using arima models. *Mathematical and computer modelling*, 51(11-12):1398–1403, 2010.

[104] Ping Han, Peng Xin Wang, Shu Yu Zhang, and De Hai Zhu. Drought forecasting based on the remote sensing data using arima models. *Mathematical and Computer Modelling*, 51(11):1398–1403, 2010.

[105] Fei Hao. The applications of markov prediction method in stock market. *Friends of Science*, 6(1):78–81, 2006.

[106] Zou Haofei, Xia Guoping, Yang Fangting, and Yang Han. A neural network model based on the multi-stage optimization approach for short-term food price forecasting in china. *Expert Systems with Applications*, 33(2):347–356, 2007.

[107] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. Biosignal data augmentation based on generative adversarial networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 368–371. IEEE, 2018.

[108] Richard Hardstone, Simon-Shlomo Poil, Giuseppina Schiavone, Rick Jansen, Vadim V Nikulin, Huibert D Mansvelder, and Klaus Linkenkaer-Hansen. Detrended fluctuation analysis: a scale-free view on neuronal oscillations. *Frontiers in physiology*, 3:450, 2012.

[109] Andrew C Harvey and PHJ Todd. Forecasting economic time series with structural and box-jenkins models: A case study. *Journal of Business & Economic Statistics*, 1(4):299–307, 1983.

[110] Hideaki Hayashi, Taro Shibanoki, Keisuke Shima, Yuichi Kurita, and Toshio Tsuji. A recurrent probabilistic neural network with dimensionality reduction based on time-series discriminant component analysis. *IEEE transactions on neural networks and learning systems*, 26(12):3021–3033, 2015.

[111] N Hemageetha and GM Nasira. Radial basis function model for vegetable price prediction. In *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pages 424–428. IEEE, 2013.

[112] HMS Chandana Herath and S McHardy. Power quality trends in energy australia distribution network. In *2008 13th International Conference on Harmonics and Quality of Power*, pages 1–6. IEEE, 2008.

[113] Stefan Hergarten. *Self organized criticality in earth systems*. Springer, 2002.

[114] Emilcy Hernández, Victor Sanchez-Anguix, Vicente Julian, Javier Palanca, and Néstor Duque. Rainfall prediction: A deep learning approach. In

International Conference on Hybrid Artificial Intelligence Systems, pages 151–162. Springer, 2016.

[115] Masahiro Higo, Sachiko Kuroda Nakada, et al. *How can we extract a fundamental trend from an economic time series?* IMES, 1998.

[116] Irma Hindrayanto, Siem Jan Koopman, and Marius Ooms. Exact maximum likelihood estimation for non-stationary periodic time series models. *Computational Statistics & Data Analysis*, 54(11):2641–2654, 2010.

[117] SL Ho, M Xie, and TN Goh. A comparative study of neural network and box-jenkins arima modeling in time series prediction. *Computers and Industrial Engineering*, 42(2):371–375, 2002.

[118] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[119] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[120] Tao Hu, Xiaoshuan Zhang, Yunxian Hou, Weisong Mu, and Zetian Fu. A hybrid model for forecasting aquatic products short-term price integrated wavelet neural network with genetic algorithm. In *International Conference on Natural Computation*, pages 352–360. Springer, 2005.

[121] Harold Edwin Hurst. Long term storage capacity of reservoirs. *ASCE Transactions*, 116(776):770–808, 1951.

[122] JT Gene Hwang and A Adam Ding. Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757, 1997.

[123] Rob Hyndman. The interaction between trend and seasonality. *International Journal of Forecasting*, 20(4):561–563, 2004.

[124] Rob J Hyndman. A brief history of forecasting competitions. *International Journal of Forecasting*, 36(1):7–14, 2020.

[125] Andrey Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[126] Girish K Jha and Kanchan Sinha. Time-delay neural networks for time series prediction: an application to the monthly wholesale price of oilseeds in india. *Neural Computing and Applications*, 24(3-4):563–571, 2014.

[127] Yan-jie Ji, Liang-peng Gao, Xiao-shi Chen, and Wei-hong Guo. Strategies for multi-step-ahead available parking spaces forecasting based on wavelet transform. *Journal of Central South University*, 24(6):1503–1512, 2017.

[128] GY Jiang. Technologies and applications of the identification of road traffic conditions. *Beijing: China Communications Press*, 11:52–53, 2004.

[129] Mikhail Kanevski, Alexei Pozdnoukhov, Alexi Pozdnukhov, and Vadim Timonin. *Machine learning for spatial environmental data: theory, applications, and software*. EPFL press, 2009.

[130] Yanfei Kang, Rob J Hyndman, and Kate Smith-Miles. Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, 33(2):345–358, 2017.

[131] Abdul Aziz Karia, Imbarine Bujang, and Ismail Ahmad. Fractionally integrated arma for crude palm oil prices prediction: case of potentially overdifference. *Journal of Applied Statistics*, 40(12):2735–2748, 2013.

[132] Monisha Kaul, Robert L Hill, and Charles Walthall. Artificial neural networks for corn and soybean yield prediction. *Agricultural Systems*, 85(1):1–18, 2005.

[133] Lars Kegel, Martin Hahmann, and Wolfgang Lehner. Feature-based comparison and generation of time series. In *Proceedings of the 30th*

International Conference on Scientific and Statistical Database Management, pages 1–12, 2018.

[134] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.

[135] Gil Keren, Nicholas Cummins, and Björn Schuller. Calibrated prediction intervals for neural network regressors. *IEEE Access*, 6:54033–54041, 2018.

[136] Mehdi Khashei and Mehdi Bijari. An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with applications*, 37(1):479–489, 2010.

[137] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356, 2011.

[138] Tae-Woong Kim and Juan B Valdes. Synthetic generation of hydrologic time series based on nonparametric random generation. *Journal of Hydrologic Engineering*, 10(5):395–404, 2005.

[139] Douglas M Kline. Methods for multi-step time series forecasting neural networks. In *Neural networks in business forecasting*, pages 226–250. IGI Global, 2004.

[140] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[141] Dimitrios Kotsakos, Goce Trajcevski, Dimitrios Gunopulos, and Charu C Aggarwal. Time-series data clustering., 2013.

[142] Vijay Kotu and Bala Deshpande. *Data Science: Concepts and Practice*. Morgan Kaufmann, 2018.

[143] Theodoros Koutroumanidis, Konstantinos Ioannou, and Garyfallos Arabatzis. Predicting fuelwood prices in greece with the use of arima

models, artificial neural networks and a hybrid arima–ann model. *Energy Policy*, 37(9):3627–3634, 2009.

[144] Thomas Kriechbaumer, Andrew Angus, David Parsons, and Monica Rivas Casado. An improved wavelet–arima approach for forecasting metal prices. *Resources Policy*, 39:32–41, 2014.

[145] Robert Krieger. *Handbook of pesticide toxicology: Principles and agents*, volume 1. Academic press, 2001.

[146] Marcin Kubacki and Janusz Sosnowski. Applying time series analysis to performance logs. In *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2015*, volume 9662, page 96623C. International Society for Optics and Photonics, 2015.

[147] Mirko Kück, Sven F Crone, and Michael Freitag. Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1499–1506. IEEE, 2016.

[148] S Vasantha Kumar and Lelitha Vanajakshi. Short-term traffic flow prediction using seasonal arima model with limited input data. *European Transport Research Review*, 7(3):21, 2015.

[149] Shiv Kumar and Cheng Hsu. An expert system framework for forecasting method selection. In *[1988] Proceedings of the Twenty-First Annual Hawaii International Conference on System Sciences. Volume III: Decision Support and Knowledge Based Systems Track*, volume 3, pages 86–95. IEEE, 1988.

[150] Fernando Sánchez Lasheras, Francisco Javier de Cos Juez, Ana Suárez Sánchez, Alicja Krzemień, and Pedro Riesgo Fernández. Forecasting the comex copper spot price by means of neural networks and arima models. *Resources Policy*, 45:37–43, 2015.

[151] Fernando Sánchez Lasheras, Francisco Javier de Cos Juez, Ana Suárez Sánchez, Alicja Krzemień, and Pedro Riesgo Fernández. Forecasting the comex copper spot price by means of neural networks and arima models. *Resources Policy*, 45:37–43, 2015.

[152] Cheng-Few Lee and John C Lee. *Handbook of financial econometrics and statistics*. Springer, 2015.

[153] Taehoon Lee, Taesup Moon, Seung Jean Kim, and Sungroh Yoon. Regularization and kernelization of the maximin correlation approach. *IEEE Access*, 4:1385–1392, 2016.

[154] Ma Lei, Luan Shiyan, Jiang Chuanwen, Liu Hongling, and Zhang Yan. A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews*, 13(4):915–920, 2009.

[155] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

[156] Christiane Lemke and Bogdan Gabrys. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016, 2010.

[157] Howard Levene. Robust tests for equality of variances. contributions to probability and statistics in olkin i, ed, 1960.

[158] Zhe-Min Li, Li-Guo Cui, Shi-Wei Xu, Ling-yun Weng, Xiao-xia Dong, Gan-Qiong Li, Hai-Peng Yu, et al. Prediction model of weekly retail price for eggs based on chaotic neural network. *Journal of integrative agriculture*, 12(12):2292–2299, 2013.

[159] Don Libes, David Lechevalier, and Sanjay Jain. Issues in synthetic data generation for advanced manufacturing. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1746–1754. IEEE, 2017.

[160] Larry S Liebovitch and Weiming Yang. Transition from persistent to antipersistent correlation in biological systems. *Physical Review E*, 56(4):4557, 1997.

[161] Lin Lin, Fang Wang, Xiaolong Xie, and Shisheng Zhong. Random forests-based extreme learning machine ensemble for multi-regime time series prediction. *Expert Systems with Applications*, 83:164–176, 2017.

[162] K Liu, S Subbarayan, RR Shoults, MT Manry, C Kwan, FI Lewis, and J Naccarino. Comparison of very short-term load forecasting techniques. *IEEE Transactions on power systems*, 11(2):877–882, 1996.

[163] Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuan-Jing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2326–2335, 2015.

[164] Changshou Luo, Qingfeng Wei, Liying Zhou, Junfeng Zhang, and Sufen Sun. Prediction of vegetable price based on neural network and genetic algorithm. In *International Conference on Computer and Computing Technologies in Agriculture*, pages 672–681. Springer, 2010.

[165] Nicolò Pietro Paolo Macciotta, Aldo Cappio-Borlino, and Giuseppe Pulina. Time series autoregressive integrated moving average modeling of test-day milk yields of dairy ewes. *Journal of dairy science*, 83(5):1094–1103, 2000.

[166] Nicolò Pietro Paolo Macciotta, Daniele Vicario, Giuseppe Pulina, and Aldo Cappio-Borlino. Test day and lactation yield predictions in italian simmental cows by arma methods. *Journal of dairy science*, 85(11):3107–3114, 2002.

[167] Nicolò Pietro Paolo Macciotta, Aldo Cappio-Borlino, and Giuseppe Pulina. Time series autoregressive integrated moving average modeling of test-day milk yields of dairy ewes. *Journal of dairy science*, 83(5):1094–1103, 2000.

[168] Thomas Maiwald, Enno Mammen, Swagata Nandi, and Jens Timmer. Surrogate data—a qualitative and quantitative analysis. In *Mathematical*

Methods in Signal Processing and Digital Image Analysis, pages 41–74. Springer, 2008.

[169] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[170] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3), 2018.

[171] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[172] Benoit B Mandelbrot and John W Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM review*, 10(4):422–437, 1968.

[173] Danilo P Mandic and Jonathon A Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley Online Library, 2001.

[174] Simona-Ioana Marinescu. Research on application of neural networks in organizational management. *Acta Universitatis Cibiniensis. Technical Series*, 65(1):64–68, 2014.

[175] David L McDowell, Jitesh Panchal, Hae-Jin Choi, Carolyn Seepersad, Janet Allen, and Farrokh Mistree. *Integrated design of multiscale, multifunctional materials and products*. Butterworth-Heinemann, 2009.

[176] Ron Meir. Nonparametric time series prediction through adaptive model selection. *Machine learning*, 39(1):5–34, 2000.

[177] Prem Melville and Raymond J Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[178] Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, Kasirat Kasfi, and Philip Smethurst. Recurrent neural networks for one day ahead prediction of stream flow. In *Proceedings of the Workshop on Time Series Analytics and Applications*, pages 25–31. ACM, 2016.

[179] Alan Miller. *Subset selection in regression*. Chapman and Hall/CRC, 2002.

[180] Wei Minghua, Zhou Qiaolin, Yang Zhijian, and Zheng Jingui. Prediction model of agricultural product's price based on the improved bp neural network. In *2012 7th International Conference on Computer Science & Education (ICCSE)*, pages 613–617. IEEE, 2012.

[181] Tang Mingming and Zhang Jinliang. A multiple adaptive wavelet recurrent neural network model to analyze crude oil prices. *Journal of Economics and Business*, 64(4):275–286, 2012.

[182] RS Mitchell, JG Michalski, and TM Carbonell. *An artificial intelligence approach*. Springer, 2013.

[183] Hamid Moeeni and Hossein Bonakdari. Forecasting monthly inflow with extreme seasonal variation using the hybrid sarima-ann model. *Stochastic Environmental Research and Risk Assessment*, pages 1–14, 2016.

[184] Hamid Moeeni and Hossein Bonakdari. Forecasting monthly inflow with extreme seasonal variation using the hybrid sarima-ann model. *Stochastic environmental research and risk assessment*, 31(8):1997–2010, 2017.

[185] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[186] Francisco Mojica, Carlos Villaseñor, Alma Y Alanis, and Nancy Arana-Daniel. Long short-term memory with smooth adaptation. In *2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–6. IEEE, 2019.

[187] Imad A Moosa. Is the export-led growth hypothesis valid for australia? *Applied Economics*, 31(7):903–906, 1999.

[188] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. Comparison of different methods for univariate time series imputation in r. *arXiv preprint arXiv:1510.03924*, 2015.

[189] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*, pages 999–1004. Springer, 1997.

[190] Mario A Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. Instance spaces for machine learning classification. *Machine Learning*, 107(1):109–147, 2018.

[191] Priya Narayanan, Ashoke Basistha, Sumana Sarkar, and Sachdeva Kamna. Trend analysis and arima modelling of pre-monsoon rainfall data for western india. *Comptes Rendus Geoscience*, 345(1):22–27, 2013.

[192] Priya Narayanan, Ashoke Basistha, Sumana Sarkar, and Sachdeva Kamna. Trend analysis and arima modelling of pre-monsoon rainfall data for western india. *Comptes Rendus Geoscience*, 345(1):22–27, 2013.

[193] Robert Nau. Notes on the random walk model. *Fuqua School of business, Duke University. 19p*, 2014.

[194] Jun Niu and Bellie Sivakumar. Scale-dependent synthetic streamflow generation using a continuous wavelet transform. *Journal of hydrology*, 496:71–78, 2013.

[195] Donald J Noakes, A Ian McLeod, and Keith W Hipel. Forecasting monthly riverflow time series. *International Journal of Forecasting*, 1(2):179–190, 1985.

[196] Rangsan Nochai and Titida Nochai. Arima model for forecasting oil palm price. In *Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications*, pages 13–15, 2006.

[197] Rangsan Nochai and Titida Nochai. Arima model for forecasting oil palm price. In *Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and applications*, pages 13–15, 2006.

[198] Ralph G O'Brien. A simple test for variance effects in experimental designs. *Psychological Bulletin*, 89(3):570, 1981.

[199] Nancy A Obuchowski. Nonparametric analysis of clustered roc curve data. *Biometrics*, pages 567–578, 1997.

[200] Amir Omidvarnia, Mostefa Mesbah, Mangor Pedersen, and Graeme Jackson. Range entropy: A bridge between signal complexity and self-similarity. *Entropy*, 20(12):962, 2018.

[201] Mahesh Pal and Paul M Mather. An assessment of the effectiveness of decision tree methods for land cover classification. *Remote sensing of environment*, 86(4):554–565, 2003.

[202] Justin D Paola and RA Schowengerdt. A review and analysis of backpropagation neural networks for classification of remotely-sensed multi-spectral imagery. *International Journal of remote sensing*, 16(16):3033–3058, 1995.

[203] Georgia Papacharalampous, Hristos Tyralis, and Demetris Koutsoyiannis. Comparison of stochastic and machine learning methods for multi-step ahead forecasting of hydrological processes. *Stochastic environmental research and risk assessment*, 33(2):481–514, 2019.

[204] Fernando De Meer Pardo and Rafael Cobo López. Mitigating overfitting on financial datasets with generative adversarial networks. *The Journal of Financial Data Science*, 2(1):76–85, 2020.

[205] Alexander G Parlos, Omar T Rais, and Amir F Atiya. Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural networks*, 13(7):765–786, 2000.

[206] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[207] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[208] Alan Zanoni Peixinho, Bárbara Caroline Benato, Luis Gustavo Nonato, and Alexandre Xavier Falcão. Delaunay triangulation data augmentation guided by visual analytics for deep learning. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 384–391. IEEE, 2018.

[209] Ali Osman Pektaş and H Kerem Cigizoglu. Ann hybrid model versus arima and arimax models of runoff coefficient. *Journal of hydrology*, 500:21–36, 2013.

[210] Diego Crespo Pereira, David del Rio Vilas, Nadia Rego Monteil, Rosa Rios Prado, and Alejandro Garcia del Valle. Autocorrelation effects in manufacturing systems performance: a simulation analysis. In *Proceedings of the 2012 winter simulation conference (WSC)*, pages 1–12. IEEE, 2012.

[211] Adhistya Erna Permanasari, Indriana Hidayah, and Isna Alfi Bustoni. Sarima (seasonal arima) implementation on time series to forecast the number of malaria incidence. In *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 203–207. IEEE, 2013.

[212] Adhistya Erna Permanasari, Dayang Rohaya Awang Rambli, and P Dhanapal Durai Dominic. Forecasting method selection using anova and duncan multiple range tests on time series dataset. In *2010 International Symposium on Information Technology*, volume 2, pages 941–945. IEEE, 2010.

[213] Fotios Petropoulos, Nikolaos Kourentzes, Konstantinos Nikolopoulos, and Enno Siemsen. Judgmental selection of forecasting models. *Journal of Operations Management*, 2018.

[214] Tuan Pham, Rob Hess, Crystal Ju, Eugene Zhang, and Ronald Metoyer. Visualization of diversity in large multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1053–1062, 2010.

[215] Raul Poler and Josefa Mula. Forecasting model selection through out-of-sample rolling horizon weighted errors. *Expert Systems with Applications*, 38(12):14778–14785, 2011.

[216] GE Powell and IC Percival. A spectral entropy method for distinguishing regular and irregular motion of hamiltonian systems. *Journal of Physics A: Mathematical and General*, 12(11):2053, 1979.

[217] Ricardo Prudêncio and Teresa Ludermir. Using machine learning techniques to combine forecasting methods. In *Australasian Joint Conference on Artificial Intelligence*, pages 1122–1127. Springer, 2004.

[218] Octavio A Ramirez and Mohamadou Fadiga. Forecasting agricultural commodity prices with asymmetric-error garch models. *Journal of Agricultural and Resource Economics*, pages 71–85, 2003.

[219] Akhter Mohiuddin Rather, Arun Agarwal, and VN Sastry. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241, 2015.

[220] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[221] Rajesh Reghunath, TR Sreedhara Murthy, and BR Raghavan. Time series analysis to monitor and assess water resources: A moving average approach. *Environmental Monitoring and Assessment*, 109(1-3):65–72, 2005.

[222] DJ Reid. A comparison of forecasting techniques on economic time series. *Forecasting in Action. Operational Research Society and the Society for Long Range Planning*, 1972.

[223] Joshua S Richman, Douglas E Lake, and J Randall Moorman. Sample entropy. In *Methods in enzymology*, volume 384, pages 172–184. Elsevier, 2004.

[224] Vicente Rico-Ramirez, Miguel A Reyes-Mendoza, Pedro A Quintana-Hernandez, Jesus A Ortiz-Cruz, Salvador Hernandez-Castro, and Urmila M Diwekar. Fisher information on the performance of dynamic systems. *Industrial & engineering chemistry research*, 49(4):1812–1821, 2010.

[225] Jorma J Rissanen. Fisher information and stochastic complexity. *IEEE transactions on information theory*, 42(1):40–47, 1996.

[226] James Sethna. *Statistical mechanics: entropy, order parameters, and complexity*, volume 14. Oxford University Press, 2006.

[227] Rudy Setiono, Karel Dejaeger, Wouter Verbeke, David Martens, and Bart Baesens. Software effort prediction using regression rule extraction from neural networks. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 45–52. IEEE, 2010.

[228] Muhammad Ammar Shafi, Mohd Saifullah Rusiman, Maria Elena Nor, Azme Khamis, Siti Nabilah Syuhada Abdullah, Mohd Syafiq Azmi, Munirah Sakinah Zainal Abidin, and Maselan Ali. The factors that influence job satisfaction among royal malaysian customs department employee. In *Journal of Physics: Conference Series*, volume 995, pages 12–16, 2018.

[229] Chandra Shah. Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting*, 13(4):489–500, 1997.

[230] Tamer Shahwan and Martin Odening. Forecasting agricultural commodity prices using hybrid neural networks. In *Computational intelligence in economics and finance*, pages 63–74. Springer, 2007.

[231] A Shamshad, MA Bawadi, WMA Wan Hussin, TA Majid, and SAM Sanusi. First and second order markov chain models for synthetic generation of wind speed time series. *Energy*, 30(5):693–708, 2005.

[232] Elnaz Sharghi, Vahid Nourani, Hessam Najafi, and Saeed Soleimani. Wavelet-exponential smoothing: a new hybrid method for suspended sediment load modeling. *Environmental Processes*, 6(1):191–218, 2019.

[233] Saeid Shokri, Mohammad Taghi Sadeghi, Mahdi Ahmadi Marvast, and Shankar Narasimhan. Integrating principal component analysis and vector quantization with support vector regression for sulfur content prediction in hds process. *Chemical Industry and Chemical Engineering Quarterly*, 21(3):379–390, 2015.

[234] Siuly Siuly, Yan Li, and Yanchun Zhang. Eeg signal analysis and classification. *IEEE Trans Neural Syst Rehabilit Eng*, 11:141–4, 2016.

[235] Eugen Slutzky. The summation of random causes as the source of cyclic processes. *Econometrica: Journal of the Econometric Society*, pages 105–146, 1937.

[236] Michael Small, Dejin Yu, and Robert G Harrison. Surrogate test for pseudoperiodic time series data. *Physical Review Letters*, 87(18):188101, 2001.

[237] Kate Smith-Miles and Simon Bowly. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63:102–113, 2015.

[238] Joshua Smolka. *Self-Learning Forecast Method Selection.* PhD thesis, Julius-Maximilians-Universität Würzburg, 2016.

[239] J Snyder, J Sweat, M Richardson, and D Pattie. Developing neural networks to forecast agricultural commodity prices. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, volume 4, pages 516–522. IEEE, 1992.

[240] Antti Sorjamaa and Amaury Lendasse. Time series prediction using dirrec strategy. In *Artificial Neural Networks, 2006. ESANN 2006. The 14th European Symposium on*, pages 143–148, Bruges, Belgium, April 2006. d-side publications.

[241] Tanya Strydom and Timothée Poisot. Svd entropy reveals the high complexity of ecological networks. 2020.

[242] KK Suresh and SR Krishna Priya. Forecasting sugarcane yield of tamilnadu using arima models. *Sugar Tech*, 13(1):23–26, 2011.

[243] KK Suresh and SR Krishna Priya. Forecasting sugarcane yield of tamilnadu using arima models. *Sugar Tech*, 13(1):23–26, 2011.

[244] Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083, 2012.

[245] Souhaib Ben Taieb, Gianluca Bontempi, Antti Sorjamaa, and Amaury Lendasse. Long-term prediction of time series by combining direct and mimo strategies. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 3054–3061. IEEE, 2009.

[246] Souhaib Ben Taieb and Rob J Hyndman. A gradient boosting approach to the kaggle load forecasting competition. *International journal of forecasting*, 30(2):382–394, 2014.

[247] Souhaib Ben Taieb, Rob J Hyndman, et al. *Recursive and direct multi-step forecasting: the best of both worlds*, volume 19. Citeseer, 2012.

[248] Souhaib Ben Taieb, Antti Sorjamaa, and Gianluca Bontempi. Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing*, 73(10-12):1950–1957, 2010.

[249] Katsuto Tanaka. A unified approach to the measurement error problem in time series models. *Econometric Theory*, 18(2):278–296, 2002.

[250] Mariusz Tarnopolski. On the relationship between the hurst exponent, the ratio of the mean square successive difference to the variance, and the number of turning points. *Physica A: Statistical Mechanics and its Applications*, 461:662–673, 2016.

[251] Stephen J Taylor. *Modelling financial time series*. world scientific, 2008.

[252] Jiashen Teh and Ian Cotton. Reliability impact of dynamic thermal rating system in wind power integrated network. *IEEE Transactions on Reliability*, 65(2):1081–1089, 2015.

[253] Marina Theodosiou. Disaggregation & aggregation of time series components: A hybrid forecasting approach using generalized regression neural networks and the theta method. *Neurocomputing*, 74(6):896–905, 2011.

[254] Felicity Thomas, Matthew Signal, and J Geoffrey Chase. Using continuous glucose monitoring data and detrended fluctuation analysis to determine patient condition: a review. *Journal of diabetes science and technology*, 9(6):1327–1335, 2015.

[255] Yin Tian, Huiling Zhang, Wei Xu, Haiyong Zhang, Li Yang, Shuxing Zheng, and Yupan Shi. Spectral entropy can predict changes of working memory performance reduced by short-time training in the delayed-match-to-sample task. *Frontiers in human neuroscience*, 11:437, 2017.

[256] Mohammad Valipour, Mohammad Ebrahim Banihabib, and Seyyed Mahmood Reza Behbahani. Comparison of the arma, arima, and the

autoregressive artificial neural network models in forecasting the monthly inflow of dez dam reservoir. *Journal of hydrology*, 476:433–441, 2013.

[257] Mohammad Valipour, Mohammad Ebrahim Banihabib, and Seyyed Mahmood Reza Behbahani. Comparison of the arma, arima, and the autoregressive artificial neural network models in forecasting the monthly inflow of dez dam reservoir. *Journal of hydrology*, 476:433–441, 2013.

[258] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[259] Peerapon Vateekul and Kanoksri Sarinnapakorn. Tree-based approach to missing data imputation. In *2009 IEEE International Conference on Data Mining Workshops*, pages 70–75. IEEE, 2009.

[260] Hrishikesh D Vinod, Javier López-de Lacalle, et al. Maximum entropy bootstrap for time series: the meboot r package. *Journal of Statistical Software*, 29(5):1–19, 2009.

[261] Jujie Wang and Yaning Li. Multi-step ahead wind speed prediction based on optimal feature extraction, long short term memory neural network and error correction strategy. *Applied Energy*, 230:429–443, 2018.

[262] Xiaozhe Wang, Kate Smith-Miles, and Rob Hyndman. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, 72(10-12):2581–2594, 2009.

[263] Yue Wang, XingYu Ye, and Yudan Huo. Prediction of household food retail prices based on arima model. In *Multimedia Technology (ICMT), 2011 International Conference on*, pages 2301–2305. IEEE, 2011.

[264] Yue Wang, XingYu Ye, and Yudan Huo. Prediction of household food retail prices based on arima model. In *2011 International Conference on Multimedia Technology*, pages 2301–2305. IEEE, 2011.

[265] Shinichi Watari. Separation of periodic, chaotic, and random components in solar activity. *Solar Physics*, 168(2):413–422, 1996.

[266] Andreas Weigend. Predicting sunspots and exchange rates with connectionist networks. *Nonlinear Model. Forecasting J.*, pages 395–432, 1992.

[267] Andreas S Weigend. *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[268] DU Wen and H Holly Wang. Price behavior in china's wheat futures market. *China economic review*, 15(2):215–229, 2004.

[269] Agus Widodo and Indra Budi. Model selection using dimensionality reduction of time series characteristics. In *International Symposium on Forecasting, Seoul, South Korea*, 2013.

[270] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[271] Herman Wold. *A study in the analysis of stationary time series*. PhD thesis, Almqvist & Wiksell, 1938.

[272] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[273] James CS Wood. Establishing and maintaining system linearity. *Current protocols in cytometry*, 47(1):1–4, 2009.

[274] CL Wu, KW Chau, and YS Li. River stage prediction based on a distributed support vector regression. *Journal of hydrology*, 358(1-2):96–111, 2008.

[275] Haoyang Wu, Huaili Wu, Minfeng Zhu, Weifeng Chen, and Wei Chen. A new method of large-scale short-term forecasting of agricultural commodity prices: illustrated by the case of agricultural markets in beijing. *Journal of Big Data*, 4(1):1–22, 2017.

[276] Yujie Wu and Jianzhou Wang. A novel hybrid model based on artificial neural networks for solar radiation prediction. *Renewable Energy*, 89:268–284, 2016.

[277] Tao Xiong, Yukun Bao, and Zhongyi Hu. Beyond one-step-ahead forecasting: evaluation of alternative multi-step-ahead forecasting models for crude oil prices. *Energy Economics*, 40:405–415, 2013.

[278] Tao Xiong, Chongguang Li, Yukun Bao, Zhongyi Hu, and Lu Zhang. A combination method for interval forecasting of agricultural commodity futures prices. *Knowledge-Based Systems*, 77:92–102, 2015.

[279] Wei-li Xiong and Bao-guo Xu. Study on optimization of svr parameters selection based on pso [j]. *Journal of System Simulation*, 9:017, 2006.

[280] Yanyan Xu, Qing-Jie Kong, Reinhard Klette, and Yuncai Liu. Accurate and interpretable bayesian mars for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2457–2469, 2014.

[281] Arpita Yadav and Kapil Sahu. Wind forecasting using artificial neural networks: a survey and taxonomy. *International Journal of Research In Science & Engineering*, 3, 2017.

[282] Qiao Yan and Cong Ma. Application of integrated arima and rbf network for groundwater level forecasting. *Environmental Earth Sciences*, 75(5):396, 2016.

[283] Liu Yi, Xu Ke, Song Junde, Zhao Yuwen, and Bi Qiang. Forecasting model based on an improved elman neural network and its application in the agricultural production. In *2013 IEEE International Conference on Granular Computing (GrC)*, pages 202–207. IEEE, 2013.

[284] Shouhua Yu and Jingying Ou. Forecasting model of agricultural products prices in wholesale markets based on combined bp neural network-time series model. In *2009 international conference on information management,*

innovation management and industrial engineering, volume 1, pages 558–561. IEEE, 2009.

[285] George Udny Yule. Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

[286] Duo Zhang, Geir Lindholm, and Harsha Ratnaweera. Use long short-term memory to enhance internet of things for combined sewer overflow monitoring. *Journal of Hydrology*, 556:409–418, 2018.

[287] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[288] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[289] G Peter Zhang and VL Berardi. Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the operational research society*, 52(6):652–664, 2001.

[290] G Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514, 2005.

[291] Gioqinang Zhang and Michael Y Hu. Neural network forecasting of the british pound/us dollar exchange rate. *Omega*, 26(4):495–506, 1998.

[292] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[293] Jin-Liang Zhang, Yue-Jun Zhang, and Lu Zhang. A novel hybrid method for crude oil price forecasting. *Energy Economics*, 49:649–659, 2015.

[294] X Zhao, A Kebbie-Anthony, S Azarm, and B Balachandran. Dynamic data-driven multi-step-ahead prediction with simulation data and sensor measurements. *AIAA Journal*, 57(6):2270–2279, 2019.

[295] Zhibiao Zhao et al. Parametric and nonparametric models and methods in financial econometrics. *Statistics Surveys*, 2:1–42, 2008.

[296] Jiajun Zong and Quanyin Zhu. Price forecasting for agricultural products based on bp and rbf neural network. In *2012 IEEE International Conference on Computer Science and Automation Engineering*, pages 607–610. IEEE, 2012.

[297] HF Zou, GP Xia, FT Yang, and HY Wang. An investigation and comparison of artificial neural network and time series models for chinese food grain price forecasting. *Neurocomputing*, 70(16):2913–2923, 2007.

[298] HF Zou, GP Xia, FT Yang, and HY Wang. An investigation and comparison of artificial neural network and time series models for chinese food grain price forecasting. *Neurocomputing*, 70(16-18):2913–2923, 2007.

Appendices

Appendix A

Significance Tests

Figure A.1: Multivariate $Tests^a$

Effect		Value	F	Hypothesis df	Error df	Sig.
time	Pillai's Trace	.649	4532.901[b]	19.000	46491.000	.000
	Wilks' Lambda	.351	4532.901[b]	19.000	46491.000	.000
	Hotelling's Trace	1.853	4532.901[b]	19.000	46491.000	.000
	Roy's Largest Root	1.853	4532.901[b]	19.000	46491.000	.000
time * Prin1	Pillai's Trace	.050	128.444[b]	19.000	46491.000	.000
	Wilks' Lambda	.950	128.444[b]	19.000	46491.000	.000
	Hotelling's Trace	.052	128.444[b]	19.000	46491.000	.000
	Roy's Largest Root	.052	128.444[b]	19.000	46491.000	.000
time * Prin2	Pillai's Trace	.163	477.478[b]	19.000	46491.000	.000
	Wilks' Lambda	.837	477.478[b]	19.000	46491.000	.000
	Hotelling's Trace	.195	477.478[b]	19.000	46491.000	.000
	Roy's Largest Root	.195	477.478[b]	19.000	46491.000	.000
time * Prin3	Pillai's Trace	.011	26.793[b]	19.000	46491.000	.000
	Wilks' Lambda	.989	26.793[b]	19.000	46491.000	.000
	Hotelling's Trace	.011	26.793[b]	19.000	46491.000	.000
	Roy's Largest Root	.011	26.793[b]	19.000	46491.000	.000
time * strategyS	Pillai's Trace	.032	80.852[b]	19.000	46491.000	.000
	Wilks' Lambda	.968	80.852[b]	19.000	46491.000	.000
	Hotelling's Trace	.033	80.852[b]	19.000	46491.000	.000
	Roy's Largest Root	.033	80.852[b]	19.000	46491.000	.000
time * method	Pillai's Trace	.036	29.734	57.000	139479.000	.000
	Wilks' Lambda	.964	29.992	57.000	138622.074	.000
	Hotelling's Trace	.037	30.249	57.000	139469.000	.000
	Roy's Largest Root	.033	80.903[c]	19.000	46493.000	.000
time * strategyS * method	Pillai's Trace	.008	6.883	57.000	139479.000	.000
	Wilks' Lambda	.992	6.898	57.000	138622.074	.000
	Hotelling's Trace	.008	6.913	57.000	139469.000	.000
	Roy's Largest Root	.008	19.291[c]	19.000	46493.000	.000

a. Design: Intercept + Prin1 + Prin2 + Prin3 + strategyS + method + strategyS * method

Within Subjects Design: time

b. Exact statistic

c. The statistic is an upper bound on F that yields a lower bound on the significance level

Figure A.2: Mauchly's Test of *Sphericity*[a]

					Epsilon[b]		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
time	.000	2664390.556	189	.000	.111	.111	.053

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. Design: Intercept + Prin1 + Prin2 + Prin3 + strategyS + method + strategyS * method

Within Subjects Design: time

b. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within Subjects Effects table.

Figure A.3: Tests of Within-Subjects Effects

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
time	Sphericity Assumed	533093.665	19	28057.561	49774.822	.000
	Greenhouse-Geisser	533093.665	2.115	251998.044	49774.822	.000
	Huynh-Feldt	533093.665	2.116	251932.118	49774.822	.000
	Lower-bound	533093.665	1.000	533093.665	49774.822	.000
time * Prin1	Sphericity Assumed	12126.879	19	638.257	1132.284	.000
	Greenhouse-Geisser	12126.879	2.115	5732.482	1132.284	.000
	Huynh-Feldt	12126.879	2.116	5730.982	1132.284	.000
	Lower-bound	12126.879	1.000	12126.879	1132.284	.000
time * Prin2	Sphericity Assumed	48060.738	19	2529.513	4487.419	.000
	Greenhouse-Geisser	48060.738	2.115	22718.732	4487.419	.000
	Huynh-Feldt	48060.738	2.116	22712.788	4487.419	.000
	Lower-bound	48060.738	1.000	48060.738	4487.419	.000
time * Prin3	Sphericity Assumed	2787.735	19	146.723	260.290	.000
	Greenhouse-Geisser	2787.735	2.115	1317.787	260.290	.000
	Huynh-Feldt	2787.735	2.116	1317.442	260.290	.000
	Lower-bound	2787.735	1.000	2787.735	260.290	.000
time * strategyS	Sphericity Assumed	9894.769	19	520.777	923.872	.000
	Greenhouse-Geisser	9894.769	2.115	4677.344	923.872	.000
	Huynh-Feldt	9894.769	2.116	4676.120	923.872	.000
	Lower-bound	9894.769	1.000	9894.769	923.872	.000
time * method	Sphericity Assumed	9003.107	57	157.949	280.206	.000
	Greenhouse-Geisser	9003.107	6.346	1418.616	280.206	.000
	Huynh-Feldt	9003.107	6.348	1418.245	280.206	.000
	Lower-bound	9003.107	3.000	3001.036	280.206	.000
time * strategyS * method	Sphericity Assumed	2394.263	57	42.005	74.517	.000
	Greenhouse-Geisser	2394.263	6.346	377.263	74.517	.000
	Huynh-Feldt	2394.263	6.348	377.164	74.517	.000
	Lower-bound	2394.263	3.000	798.088	74.517	.000
Error(time)	Sphericity Assumed	498116.361	883671	.564		
	Greenhouse-Geisser	498116.361	98388.277	5.063		
	Huynh-Feldt	498116.361	98414.023	5.061		
	Lower-bound	498116.361	46509.000	10.710		

Figure A.4: Tests of Within-Subjects Effects, cont.

Figure A.5: Tests of Between-Subjects Effects

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Intercept	1754643.543	1	1754643.543	437944.174	.000
Prin1	13382.419	1	13382.419	3340.138	.000
Prin2	22401.669	1	22401.669	5591.267	.000
Prin3	7090.525	1	7090.525	1769.735	.000
strategyS	1136.737	1	1136.737	283.720	.000
method	34678.883	3	11559.628	2885.185	.000
strategyS * method	1854.230	3	618.077	154.267	.000
Error	186340.455	46509	4.007		

Figure A.6: 6. strategyS * method * time

Measure: MEASURE_1

strategyS	method	time	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
MRFA	NN	1	-7.658[a]	.038	-7.732	-7.584
		2	-7.134[a]	.031	-7.195	-7.072
		3	-6.869[a]	.030	-6.927	-6.810
		4	-6.678[a]	.029	-6.735	-6.622
		5	-6.506[a]	.028	-6.561	-6.450
		6	-6.358[a]	.028	-6.413	-6.303
		7	-6.223[a]	.028	-6.278	-6.169
		8	-6.107[a]	.028	-6.161	-6.054
		9	-6.002[a]	.027	-6.056	-5.949
		10	-5.903[a]	.027	-5.956	-5.850
		11	-5.811[a]	.027	-5.864	-5.759
		12	-5.726[a]	.027	-5.779	-5.674

		13	-5.648[a]	.027	-5.700	-5.596
		14	-5.577[a]	.026	-5.629	-5.525
		15	-5.510[a]	.026	-5.562	-5.458
		16	-5.445[a]	.026	-5.497	-5.394
		17	-5.386[a]	.026	-5.437	-5.335
		18	-5.329[a]	.026	-5.380	-5.278
		19	-5.274[a]	.026	-5.325	-5.223
		20	-5.222[a]	.026	-5.272	-5.171
	RNN	1	-7.455[a]	.038	-7.528	-7.381
		2	-7.033[a]	.031	-7.095	-6.972
		3	-6.813[a]	.030	-6.871	-6.754
		4	-6.647[a]	.029	-6.703	-6.590
		5	-6.486[a]	.028	-6.541	-6.430
		6	-6.345[a]	.028	-6.400	-6.290
		7	-6.214[a]	.028	-6.269	-6.160
		8	-6.100[a]	.027	-6.154	-6.046
		9	-5.991[a]	.027	-6.045	-5.938
		10	-5.888[a]	.027	-5.941	-5.835
		11	-5.792[a]	.027	-5.845	-5.739
		12	-5.703[a]	.027	-5.755	-5.651
		13	-5.619[a]	.027	-5.671	-5.567
		14	-5.542[a]	.026	-5.594	-5.490
		15	-5.469[a]	.026	-5.521	-5.417
		16	-5.399[a]	.026	-5.451	-5.348
		17	-5.336[a]	.026	-5.387	-5.285
		18	-5.275[a]	.026	-5.326	-5.224
		19	-5.217[a]	.026	-5.268	-5.166
		20	-5.161[a]	.026	-5.211	-5.110
	SVR	1	-6.265[a]	.038	-6.338	-6.191
		2	-5.860[a]	.031	-5.921	-5.798
		3	-5.648[a]	.030	-5.707	-5.590
		4	-5.481[a]	.029	-5.538	-5.425
		5	-5.326[a]	.028	-5.381	-5.270
		6	-5.189[a]	.028	-5.244	-5.134
		7	-5.068[a]	.028	-5.123	-5.014
		8	-4.963[a]	.027	-5.017	-4.909
		9	-4.866[a]	.027	-4.919	-4.812
		10	-4.777[a]	.027	-4.830	-4.724

Figure A.7: 6. strategyS * method * time, cont.

		11	-4.696[a]	.027	-4.748	-4.643
		12	-4.621[a]	.027	-4.673	-4.569
		13	-4.553[a]	.027	-4.605	-4.500
		14	-4.490[a]	.026	-4.542	-4.439
		15	-4.434[a]	.026	-4.485	-4.382
		16	-4.380[a]	.026	-4.431	-4.329
		17	-4.330[a]	.026	-4.381	-4.279
		18	-4.284[a]	.026	-4.335	-4.233
		19	-4.241[a]	.026	-4.292	-4.190
		20	-4.200[a]	.026	-4.250	-4.149
REC	NN	1	-9.211[a]	.038	-9.285	-9.137
		2	-8.413[a]	.031	-8.475	-8.352
		3	-8.035[a]	.030	-8.093	-7.977
		4	-7.767[a]	.029	-7.823	-7.710
		5	-7.540[a]	.028	-7.596	-7.485
		6	-7.359[a]	.028	-7.413	-7.304
		7	-7.192[a]	.028	-7.247	-7.138
		8	-7.048[a]	.027	-7.102	-6.994

Figure A.8: 6. strategyS * method * time, cont.

	9	-6.915[a]	.027	-6.968	-6.861
	10	-6.793[a]	.027	-6.846	-6.740
	11	-6.679[a]	.027	-6.732	-6.626
	12	-6.572[a]	.027	-6.625	-6.520
	13	-6.474[a]	.027	-6.526	-6.422
	14	-6.387[a]	.026	-6.439	-6.335
	15	-6.301[a]	.026	-6.353	-6.250
	16	-6.222[a]	.026	-6.274	-6.171
	17	-6.147[a]	.026	-6.198	-6.096
	18	-6.077[a]	.026	-6.128	-6.026
	19	-6.009[a]	.026	-6.060	-5.959
	20	-5.946[a]	.026	-5.996	-5.895
RNN	1	-8.441[a]	.038	-8.515	-8.367
	2	-7.655[a]	.031	-7.717	-7.594
	3	-7.202[a]	.030	-7.260	-7.144
	4	-6.898[a]	.029	-6.954	-6.841
	5	-6.631[a]	.028	-6.687	-6.575
	6	-6.413[a]	.028	-6.468	-6.358
	7	-6.213[a]	.028	-6.267	-6.159
	8	-6.043[a]	.027	-6.097	-5.989
	9	-5.885[a]	.027	-5.938	-5.831
	10	-5.746[a]	.027	-5.799	-5.693
	11	-5.614[a]	.027	-5.666	-5.561
	12	-5.497[a]	.027	-5.549	-5.445
	13	-5.385[a]	.027	-5.437	-5.333
	14	-5.284[a]	.026	-5.336	-5.233
	15	-5.188[a]	.026	-5.240	-5.137
	16	-5.101[a]	.026	-5.152	-5.050
	17	-5.015[a]	.026	-5.067	-4.964
	18	-4.937[a]	.026	-4.988	-4.886
	19	-4.860[a]	.026	-4.911	-4.809
	20	-4.791[a]	.026	-4.841	-4.740
SVR	1	-6.463[a]	.038	-6.536	-6.389
	2	-5.950[a]	.031	-6.011	-5.888
	3	-5.675[a]	.030	-5.733	-5.617
	4	-5.470[a]	.029	-5.527	-5.414
	5	-5.289[a]	.028	-5.345	-5.234
	6	-5.143[a]	.028	-5.198	-5.088

Figure A.9: 6. strategyS * method * time, cont.

7	-5.012[a]	.028	-5.067	-4.958
8	-4.901[a]	.027	-4.955	-4.847
9	-4.800[a]	.027	-4.854	-4.747
10	-4.709[a]	.027	-4.763	-4.656
11	-4.627[a]	.027	-4.679	-4.574
12	-4.551[a]	.027	-4.604	-4.499
13	-4.482[a]	.027	-4.534	-4.430
14	-4.419[a]	.026	-4.471	-4.367
15	-4.360[a]	.026	-4.412	-4.309
16	-4.306[a]	.026	-4.357	-4.254
17	-4.255[a]	.026	-4.306	-4.204
18	-4.207[a]	.026	-4.258	-4.156
19	-4.162[a]	.026	-4.213	-4.111
20	-4.120[a]	.026	-4.170	-4.069

a. Covariates appearing in the model are evaluated at the following values: Prin1 = .009839256388, Prin2 = -.009464640973, Prin3 = -.001309757634.

Figure A.10: 6. strategyS * method * time, cont.

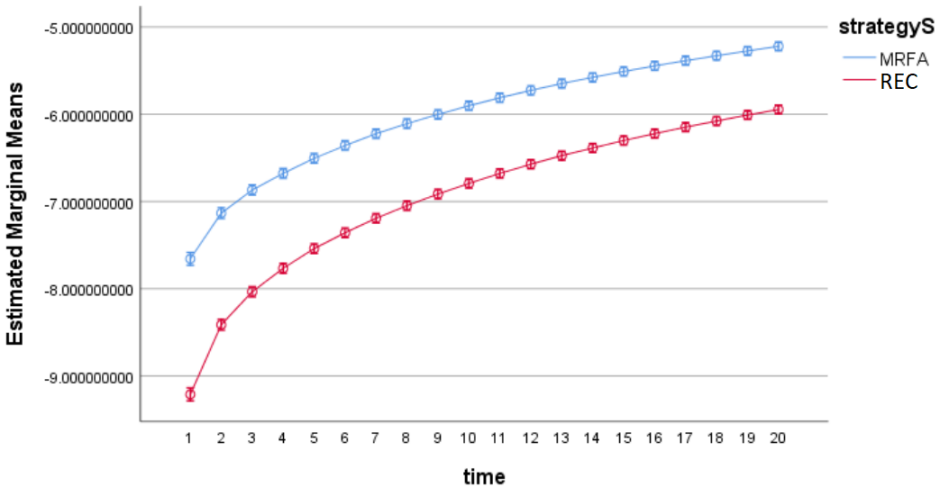


Figure A.11: Estimated Marginal Means of $MEASURE_1$ at method=NN

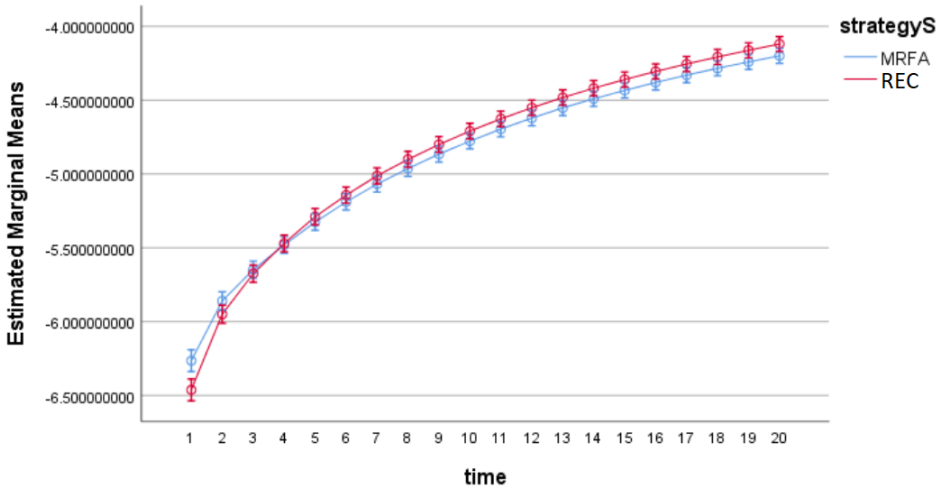


Figure A.12: Estimated Marginal Means of $MEASURE_1$ at method=SVR

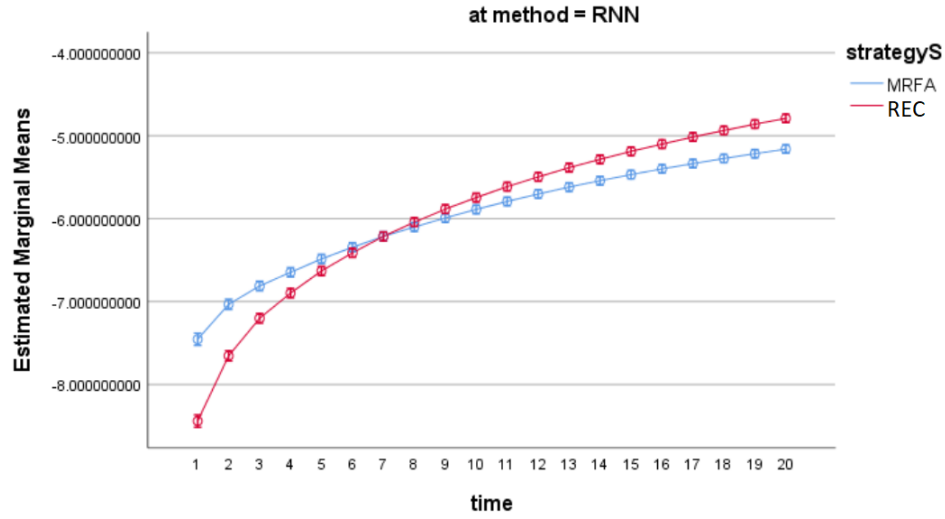


Figure A.13: Estimated Marginal Means of $MEASURE_1$ at method=RNN